(2)

MATCHING AND VERTEX PACKING: HOW "HARD" ARE THEY?

by

Michael D. Plummer*
Department of Mathematics
Vanderbilt University
Nashville, Tennessee 37235, USA

ABSTRACT

Two of the most well-known problems in graph theory are:

(a) Find a maximum matching (or perfect matching, if one exists), and

(b) Find a maximum independent set of vertices.

The first problem—usually called the *matching problem*—is known to have a polynomial algorithm; the second—often called the *vertex packing problem*—is known to be NP-complete. However, many graph theorists—especially those who do not deal much with complexity of algorithms—know little more about the complexity issues associated with these two problems than these two basic facts.

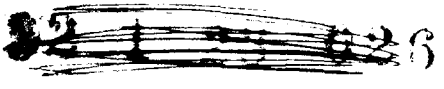
What is not so widely known within the graph theory community is that these two problems have motivated a great deal of recent activity in the area of algorithms and their complexity. Of course it is not known whether or not P = NP, but most workers in the area currently believe that equality is unlikely to hold. Motivated by this belief, a number of people have studied variations of both matching and vertex packing with the general theme being two-fold. On the one hand, one can add various side conditions to the matching problem and study the complexity–both sequential and parallel–of the resulting problems. On the other hand, one can investigate certain large and interesting classes of graphs trying to prove that for these classes the vertex packing problem has a polynomial solution.

Each branch of this two-pronged attack has yielded both interesting theorems and perplexing unsolved problems. This paper will survey this work.

1 Introduction and Background: The two fundamental Problems

In this paper, graphs will be assumed to be connected, undirected and will have no multiple edges or loops. A **matching** is any set of independent edges; i.e., no two have a vertex in common. A **maximal** matching is a matching not properly contained in any other matching. A **maximum** matching is one of largest cardinality. A **perfect** matching (sometimes called a **1-factor**) in a graph G is a matching which covers all vertices of G. A set of vertices $S \subseteq V(G)$ is **independent** if no two vertices of S are adjacent. An

* work supported by ONR Contract #N00014-85-K-0488, #N00014-91-J-1142 and Laboratoire de Recherche en Informatique, CNRS, Univ. Paris Sud



independent set S is **maximal** if it is not a proper subset of any other independent set and **maximum** if it is an independent set of largest cardinality.

So far, then, maximal and maximum matchings and independent sets are quite analogous; matchings corresponding to sets of edges and independent sets corresponding to sets of vertices. To be sure, they are related. A (maximal, maximum) matching in a graph G corresponds to a (maximal, maximum) independent set in the line graph $L(G)$. But we shall soon see that the concepts quickly diverge in difficulty.

First, however, let us note that matching and vertex packing remain closely related, at least in the computational sense, if one considers the class of *bipartite* graphs. Historically speaking, it was this class of graphs which was first studied with the task in mind of finding maximum matchings and independent sets. Implicit in the early work of Kõnig and Egerváry were the roots of the first bipartite matching algorithms. (For much more complete histories of matching algorithms see [1] and [2].) Using classical alternating path (or slightly more efficiently, alternating tree) arguments, one can find an algorithm to find a maximum matching in a bipartite graph. Moreover, the algorithm provides a maximum matching in a number of steps *polynomial* in the size of the input encoding of the graph. (See the next section for more information on polynomial and other types of complexity.)

As an important bonus, however, these classical bipartite matching algorithms also yield a **minimum vertex cover** of the graph; i.e., a smallest set of vertices which collectively contain at least one endvertex of every edge in the graph. One of the classical results of bipartite matching and covering due to Kõnig [3,4] says that the size of any maximum matching equals the size of any minimum vertex cover. At this point, one need only observe the simple, but important, fact that the complement of a (minimum) vertex cover must be a (maximum) independent set of vertices (cf. Gallai [5]) and presto! We have a *polynomial* algorithm for vertex packing.

For graphs which are *not* bipartite, however, the situation changes dramatically. Computationally the two roads of matching and vertex packing diverge quickly. We will follow the route of matching first.

Although the (polynomial) roots of matching algorithms for bipartite graphs date to the 1930's with Kõnig–Egerváry, it was not until 1965 that the first polynomial algorithm for graphs in general was found by Edmonds [6]. Edmonds himself gave an implementation in $O(n^4)$ time. (As we write this paper, Blum [7,8,9] claims that the Edmonds' implementation can be shortened to $O(n^3)$ without complications.) Since Edmonds' original paper, there have been a number of papers successively reducing the time bound.

For the past ten years or so, Micali and V.V. Vazirani [10] have held the record of $O(\sqrt{n}m)$ time. Interestingly, the first proof of correctness of this algorithm did not appear until 1989 [11] and is some forty pages in length!

It is interesting to note that again until very recently the bound for *bipartite* graphs has been no better than that for general graphs, although Hopcroft and Karp [12,13] attained the $O(\sqrt{n}m)$ bound some years earlier than did Micali and Vazirani. This bound stood until earlier this year when new breakthroughs in bipartite matching were made by the quartet of Alt, Blum, Mehlhorn and Paul, and, independently, by Feder and Motwani [14]. The first four authors [15] improved the time bound to $O(n^{1.5}\sqrt{m/\log n})$ using a new "fast adjacency matrix scanning technique" due to Cheriyan, Hagerup and Mehlhorn

[16]. This bound improves the old time bound by a factor of $\sqrt{\log n}$ when the graph is dense—i.e., when the number of edges is $O(n^2)$. Feder and Motwani, on the other hand, developed an algorithm which is even faster. Their algorithm finds a maximum matching in a bipartite graph in time $O(\frac{\sqrt{n}m}{\kappa(n,m)})$ where $\kappa(n,m) = (\log n)/[\log(n^2/m)]$.

But now let us return to the fork in the road and discuss vertex packing for graphs in general. The best deterministic sequential algorithm known to the author is $O(2^{n/3})$ due to Tarjan and Trojanowski [17] which clearly shows the large gap presently extant between matching and vertex packing.

Consider the special case when G is the line graph of another graph, a situation which was observed above to guarantee a polynomial algorithm for vertex packing in G. It is well-known that line graphs have a characterization in terms of a list of nine forbidden induced subgraphs (cf. Harary [18]). Of these nine subgraphs, perhaps the most widely studied is the *claw* $K_{1,3}$. A graph containing no induced claw is said to be **claw-free**. In 1980, Minty [19] and Sbihi [20] independently proved that if a graph is claw-free, then the vertex packing problem can be solved in polynomial time.

The remainder of this paper will be organized as follows. In Section 2 we present a rather intuitive survey of the complexity classes in which we will be primarily interested, together with an illustration showing the known containment relationships among the classes. Section 3 contains a number of variations on matching which have been formulated and the status of their complexity, if known. Section 4 deals with vertex packing variants and relations in much the same way that Section 3 deals with matching. Sections 5 and 6 deal with matching and vertex packing in parallel, respectively. Section 7 deals with the status of the matching and vertex packing problems involving counting, both exact and approximate. Finally, Section 8 is a short introduction to the work on *lower* bounds for the complexity of matching and vertex packing, especially that of Razborov.

We conclude with a list of over 260 references.

2. Complexity Classes

So far we have spoken only about algorithms which are *polynomial*. But what does this mean precisely? What kind of computational devices are being used? Does it matter? What is "NP" anyway? In order not to get bogged down, we will, for the most part, unashamedly sidestep these important issues, trying instead to present our complexity results at a more intuitive level. Fortunately, we have excellent resources to fall back on. The "bible" of complexity theory remains the book of Garey and Johnson [21], together with Johnson's ongoing guide in the Journal of Algorithms. This author, himself a neophyte in the jungles of complexity theory, feels compelled to admit that after giving the conference talk which corresponded to a rough first draft of this paper, soon found himself spending nearly a year in the subsequent ferreting out of such things as what a "Monte Carlo" algorithm truly is (not everyone quite agrees on the definition, it seems!) and, most of all, trying to compile the table of complexity classes and the corresponding lattice of containments shown below. With immense feelings of relief (mixed with not a little irony) after we had suffered through most of this, we discovered the newly published "Handbook of Theoretical Computer Science Volume A: Algorithms and Complexity". Most of our

lattice can be gleaned from the collection of lattices provided by Johnson in Chapter 2 of this bock [22]. In general, four chapters in this book are excellent sources for other facets of our survey. We refer our readers to Chapter 1 [23] by van Emde Boas for machine models, Chapter 2 [22] by Johnson on complexity classes and their interrelationships, Chapter 14 by Boppana and Sipser [24] for the *Boolean circuit* approach to complexity and to Chapter 17 [25,26] by Karp and Ramachandran for parallel computation.

Other good references on various aspects of complexity that were especially helpful to the author include [27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50, 51,52,53,54,55,56].

The types of problems of concern to us come in different varieties. (1) *decision* problems—often called simply "yes–no" problems ("Does graph G have a perfect matching?"); (2) *search* problems ("Given graph G, find a maximum independent set of vertices in G."); and (3) *counting* problems ("Given a graph G, compute $\Phi(G)$, the number of perfect matchings in G.").

This would seem to be an appropriate place to be more precise about decision versions of our two fundamental problems. In particular the "yes–no" variants of matching and vertex packing which we shall most often refer to are:

(1) Given graph G and $k > 0$, does G have a matching of size $\geq k$?

(2) Given graph G and $k > 0$, does G have an independent set of size $\geq k$?

We will call problem (1) the *matching problem* and denote it by **MATCH**. Problem (2) will be called the *vertex packing problem* and will be referred to as **VP**. Sometimes others have been known to call the *perfect* matching problem by our term MATCH, but this should cause us no problems in this paper.

Now.....as to complexity classes. In Section 1, we began our discussion with the matching problem, a problem known to have a polynomial solution. A problem is in class **P** if there is an algorithm to solve it which runs in time polynomial in the size of the input function. In other words, the algorithm *always* terminates in a number of steps polynomial in input size.

Historically, it is fair to say, one of the most important roots of computational complexity theory is the concept of the class **NP**. These are the problems which can be solved in *non-deterministic* polynomial time. We will mostly avoid the treatment of nondeterministic *machines* and *circuits* by instead explaining NP via the "certificate" method. The NP machine is allowed to "guess" a solution to a given problem (or "consult an oracle"), but then must provide a certificate for the solution which can be checked in polynomial time. For example, the problem "Does graph G contain a Hamilton cycle?" is in class NP, for given an H-cycle in G, one can convince himself/herself that it is indeed a Hamilton cycle. (Just draw it! That is, output it as an alternating sequence of vertices and edges and then check the sequence to be sure that all the vertices of $V(G)$, except exactly one, appear once, and this one exception (the "beginning" and "end" vertex) appears exactly twice. Finally, check that each edge in the sequence joins the vertex preceding it in the sequence to the vertex succeeding it.)

It is crucial to understanding the concept of NP that we realize that how the Hamilton cycle was found in the first place is irrelevant; we are only interested in certifying in

polynomial time that it is indeed a Hamilton cycle.

Now what if the input graph G does *not* have a Hamilton cycle? No "certificate" of this fact in the above sense is known. The opposite "yes–no" situation occurs for the following question: "Is every maximal independent set in graph G of the same size?". (Such graphs are called **well-covered** or **w-c** in short and were introduced in [57].) This time if the answer is "no", it is easy to certify. Just exhibit two maximal independent sets of different size. Note that maximality is easy to check. But if the answer is "yes", there is no known certification.

Define the class **co-NP** to be the class of all "yes–no" problems such that if the answer is "no", it can be certified in polynomial time. So the Hamilton cycle problem is in NP and the well-covered graph problem is in co-NP.

Clearly, P $\subseteq$ NP $\cap$ co-NP. Does equality hold? Is P = NP? Is NP = co-NP? All are open questions.

Next let us try to formalize the idea of a "hardest" problem in a complexity class. We will be content to illustrate with class NP. We begin with the notion of a *polynomial transformation*. If π and π' are decision problems, a **polynomial transformation** from π to π' is a function f from inputs x of π to inputs $f(x)$ of π' such that whenever x yields the answer "yes" for problem π, $f(x)$ yields the answer "yes" for problem π'. Moreover, the (deterministically-computable) function $f(x)$ can be computed in polynomial time.

The complexity theorist would hasten to add that the concept of a (polynomial) transformation should be contrasted with the more general notion of a *polynomial (Turing) reduction* which allows multiple calls of a subroutine. However, we shall not refer to the latter type of reduction here.

A problem π is **NP-hard** if, given any problem π' in NP, there is a *polynomial* transformation of every instance of π' to an instance of problem π. That is, if we can solve π in polynomial time, then we can solve π' polynomially as well. If, in addition, problem π belongs to NP, we say that π is **NP-complete**. That is to say, it is a "hardest" problem in NP. The first NP-complete problem was found by Cook [58] in 1971. It is the satisfiability problem of logic (or "SAT" for short). Suppose formula F is the conjunction of a set of disjunctions of propositional variables. Is there an assignment of trues and falses to the literals making F true?

Levin [59] (see also [60] in which a corrected translation of Levin's article appears as an appendix) independently and essentially simultaneously arrived at the notion of NP-completeness.

Shortly after Cook's ground-breaking result, Karp [61] published a list of 21 additional NP-complete problems. One of these was VERTEX COVER defined as follows. "Given a graph G and an integer $k > 0$, does G have a set of k or fewer vertices which collectively touch all the edges of G?" Since the complement of a vertex cover is an independent set, VERTEX COVER is clearly equivalent to the vertex packing problem VP defined earlier.

It makes sense to ask if there are "complete" problems for other complexity classes as well. For example, for class P. One must be careful, though, to carefully define the allowable types of transformations. For example, with polynomial transformations allowed, *every* problem in P would be "complete"! This, then, hardly captures a useful meaning for a "hardest" problem in P. In the case of class P, a different kind of transformation

is employed—so-called *log-space* transformations. In terms of a worktape of a Turing machine, a **log-space** transformation is one that is computable in logarithmic space on the worktape. Note that these transformations are also polynomial transformations since a logarithmic workspace bound means that only a polynomial number of distinct states on the worktape are possible.

In this more restricted sense of log-space transformation, class P still has complete problems; probably the most famous of these is the *linear programming problem* [62]. There is, however, another P-complete problem closer in spirit to our paper. This is LFMAXLVP—lexicographically first maximal independent set [63,64]. Label the vertices in the graph with the integers $1, 2, \ldots, n$ and begin to build a maximal independent set I by putting vertex 1 into I, deleting $\{1\} \cup N(1)$, where $N(1)$ denotes the set of neighbors of the vertex 1. Select the vertex with smallest label remaining and put it into set I. Delete it and all its neighbors and continue in this manner until all vertices have either been included in I or discarded. The "yes–no" question is then: "Is vertex n in set I?". (Throughout this paper, n will refer to the number of vertices in a graph and m to the number of edges, unless otherwise stated.)

Another question close to home for us in this paper is: "Is the perfect matching problem P-complete?" The evidence so far leads most workers active in the area to conjecture that the answer is "no", but the problem remains open. (Cf. [63,64].)

With these basic locations on our lattice behind us, let us continue to discuss some additional classes. In fact, let us consider parallel computation classes next. A problem is said to be in parallel class $\mathbf{NC}^i$ if it can be solved in $O((\log n)^i)$ time with a polynomial number of processors. Thus i is the degree of the time-bounding polynomial in $\log n$. The absence of an index on the processor bound in naming this class would seem to support the contention that "time is money" and "processors are cheap"! To be fair, it should be emphasized that many complexity theorists concern themselves with the time versus hardware tradeoff by considering the product of the time and processor bounds as the "true" measure of efficiency.

The class $\mathbf{NC} = \cup_{i=1}^{\infty} \mathbf{NC}^i$ is called "Nick's class" in honor of its first proponent, Nick Pippenger [65].

We shall consider our parallel processing to be done on a **CREW PRAM**, a *Concurrent-Read Exclusive Write Parallel Random Access Machine*. In such a device, two processors may read the contents of the same memory cell at the same time, but only one at a time may write in such a cell. Some of the results mentioned later in the sections devoted to parallel problems may refer to the more restricted family of EREW PRAM's where the acronym is self-explanatory. However, the differences which arise between these two models usually mean only a difference in the "i" part of NC^i and hence no difference in whether or not a given problem is in class NC. For this reason the class NC is often referred to as being "robust".

Although we will have much less to say about it, we would be remiss if we did not at least mention the Boolean circuit model of parallel computing. In this model the class NC^i is defined as a set of *languages* (i.e. subsets of $\{0,1\}^*$) recognizable (i.e., "accepted") by classes of Boolean circuits having a polynomial number of "and", "or" and "not" gates and having depth $O((\log n)^i)$. (These circuits can be thought of as directed acyclic graphs with

certain vertices designated as *inputs* (indegree = 0) and a special *output* vertex (outdegree = 0). The **depth** of the circuit is the length of any longest path from an input to the output and is the circuit analog of "parallel time". In addition, the class of Boolean circuits is usually assumed to be *log-space uniform*. (A class of Boolean circuits $\{B_n\}_1^\infty$ is said to be **log-space uniform** if there is a deterministic Turing machine which, for each n, can construct circuit B_n in space $O(\log n)$. (See [22] and [24].)

In summary, then, we have $\mathrm{NC}^1 \subseteq \mathrm{NC}^2 \subseteq \cdots \mathrm{NC} \subseteq \mathrm{P}$. Whether or not equality holds at any stage of this chain is unknown. We have here at the onset still another unsettled question about matching. Does the perfect matching problem lie in NC?

Let us next give a brief introduction to *randomized* complexity classes. The main motivation here is the following. It may be the case that there is an algorithm for a certain problem which gives the correct answer most—but not all—of the time. However, its execution time is faster than some deterministic alternative. In other words, it may be possible (and often desirable) to trade some accuracy for speed.

Let us begin with the random complexity class **RP** (for "random" polynomial time). These are the problems for which there exist algorithms which behave as follows. If the correct answer to an input is "yes", the algorithm returns the answer "yes" with probability $\geq p$, where p is some fixed probability bounded away from 0. (Often this probability is given the value 1/2 in this definition, but any fixed positive value will do.) However, if the correct answer is "no", the algorithm returns the incorrect answer "yes" with probability = 0. Moreover, it does this in polynomial time. Such algorithms are said to have "one-sided error". In other words, if the algorithm ever answers "yes", one can be certain that "yes" is indeed the correct answer!

The reader is warned that some authors call *all* randomized algorithms *Monte Carlo*, while others reserve this term for those algorithms with one-sided error, like those in class RP.

Let us also briefly mention a class containing RP, the class **BPP**. (The "B" stands for error *bounded away* from 1/2.) Members of this class are those problems with polynomial algorithms which return the correct answer (be it "yes" or "no") with probability $\geq 1/2+\epsilon$, for some $\epsilon > 0$. For this reason, BPP is called a "two-sided error" class. For a BPP problem one can rerun a given input a number of times and take the majority answer. The more runs the more probable that the majority answer is correct.

Finally, we mention the randomized complexity class **ZPP**. (The "Z" stands for "zero error".) Here the algorithm either gives the correct answer (be it "yes" or "no") in polynomial time or it refuses to answer at all! Moreover, the probability of no answer at all is less than 1/2. Thus, while BPP and RP algorithms may "lie", a ZPP algorithm never does! Randomized algorithms which do not lie are called *Las Vegas* algorithms, a term due to L. Babai.

There are, of course, *parallel* analogs to all of these randomized classes. Of these we shall be concerned in this paper only with **RNC** (and one lonely example in **ZNC**!). Clearly, $\mathrm{NC} \subseteq \mathrm{RNC}$ by definition, but whether or not equality holds is another important open question.

Now let us turn to a problem of a different nature. To set the stage, let us backtrack a bit. In defining all the complexity classes up to this point, we have dealt exclusively with

decision problems; i.e., problems which have answers which are either "yes" or "no". Of course there are other types of problems which we would like to consider. *Search* problems, for example, are especially central to this paper. Examples central to our theme are: given a graph "find the size of a maximum matching" or "find the size of a maximum independent set". In more extensive treatments of complexity (cf. again [22]) such distinctions are discussed at length and even separate complexity classes are defined to reflect these differences. For example, class **FP** ("F" is for "function") is defined as the search analog of decision class P. But even more often in the existing literature, such distinctions are glossed over. In the interests of efficiency in attaining the goals of this survey paper while the reader is still awake, we shall also ignore such distinctions for the most part.

Having said that, however, we next treat yet a third type of problem of considerable interest to graph theorists. These are the so-called *counting problems*. To be sure, such problems are "functional", but the output in this case is a non-negative integer. A nondeterministic Turing machine which, given an input string x, outputs $f(x)$ = the number of accepting computations for this input string, is called a **counting** Turing machine. The class **#P** is defined to be the set of all functions that are computable by polynomial time counting Turing machines.

It is perhaps most instructive to think of a #P function as one with the "magical" property that it *instantly* prints out the *number* of acceptable computations of an associated polynomial time non-deterministic Turing machine.

Some familiar examples of #P problems are:

(a) Given graph G, how many Hamilton cycles does it contain?

(b) Given graph G, how many 3-colorings of its vertices are there?

(c) Given graph G, how many perfect matchings does it contain?

A problem is **#P-hard** if there are polynomial time Turing reductions from all problems in #P to the given problem. If, in addition, the problem belongs to the class #P as well, then we say that the problem is **#P-complete**.

The main new idea is that for #P, the polynomial transformations from problem to problem are required to be *parsimonious*, that is, they must preserve the *number* of solutions. Valiant [66,67] showed that the problem of the number of perfect matchings in a graph (we shall call it **#PM**) is #P-complete and therefore NP-hard, even when the graph is *bipartite*. This problem is the main reason for introducing the class #P in the present paper.

Strictly speaking, it does not make sense to ask if "NP is a subclass of #P", since the former is a set of languages (a language in turn is just a set of strings), but #P is a set of functions from strings to the non-negative integers. Nevertheless, in our Framework we have drawn a dashed arrow from NP∪co-NP to #P to indicate that problems in #P are harder by their very nature to compute then are those in NP∪co-NP.

For example, a #P-machine will compute the number of Hamilton cycles in a graph G. If that number equals 0, then G has no Hamilton cycle, but if that number is greater than 0, it does. Thus we have an *instant* answer to a proven NP-complete problem!

Here we bring a halt to our discussion of complexity classes. The reader likely will want to refer to our "Framework" illustration to see the containment relations known to

exist (at this time anyway) among the various classes. (See [22] for those classes we do not discuss here.)

Finally, we will defer definitions and discussion of the two classes D^P and CC until Sections 4 and 5 respectively where we encounter them for the first and only times in this paper.

The reader will note that we have drawn a horizontal dashed line across the Framework diagram. Those classes below the line are those known to be polynomial time computable (where in some of these classes recall that randomization is allowed).

3. Matching Variations and their Complexity

We now give a list of some variants on the standard matching problem and give their complexity, if known.

Recall that the matching problems—both "perfect" and "maximum" varieties—are known to be in P. An even simpler problem (at least sequentially) is the problem of finding a maxim*al* matching. The greedy algorithm will solve this nicely. Choose a edge and delete its endvertices. Choose a edge in the remaining graph and delete its endvertices. Continuing in this manner we have a trivially polynomial (sequential) algorithm for maximal matching.

It is perhaps surprising then that the following problem (MINIMUM MAXIMAL MATCHING) is NP-complete.

1. Given graph G and integer $k > 0$, does G have a maximal matching of size $\leq k$?

NP-completeness was first shown by Yannakakis and Gavril [68]. In fact, they proved NP-completeness even in the cases where G has maximum degree at most 3 and is either planar or bipartite. The same two authors found an $O(n)$ algorithm for the problem in the case when the graph G is a tree.

Chronologically, one of the first generalizations of matching to be shown to be NP-complete was the *3-dimensional matching problem* (**3DM**). This is most easily described as follows. Let H be a 3-uniform hypergraph (i.e., each "edge" contains three vertices, not two).

2. Given H, is there a perfect matching of hyperedges, i.e., a set of hyperedges such that each vertex of H lies in exactly one hyperedge?

This problem was one of Karp's original 21 [61] and the proof is by reduction from 3SAT.

The NP-completeness of 3DM easily implies that the following two matching problems are NP-complete (see [69]).

3. Given a bipartite graph G and a partition $E(G) = E_1 \cup \cdots \cup E_k$ of its edges, does G contain a perfect matching F such that either $E_i \subseteq F$ or $E_i \cap F = \emptyset$, for all $i = 1, \ldots, k$?

4. Given a bipartite graph G and a coloring of its edge set $E(G)$, does G contain a perfect matching with exactly one edge of each color?

Motivated by the fact that certain timetable and image analysis problems can be modeled as matching problems with certain additional restrictions, Itai, Rodeh and Tanimoto [70] introduced the RESTRICTED MATCHING problem.

5. Given a bipartite graph G, a collection $R_1, \ldots, R_k$ of subsets of $E(G)$ and a collection of nonnegative integers $r_1, \ldots, r_k$, does G have a perfect matching F such that $|F \cap R_i| \leq r_i$ for $i = 1, \ldots, k$?

If k is set equal to 1 in problem 5, the resulting problem is equivalent to finding a perfect matching using as few edges of R_1 as possible and this is a very special case of the minimum weight perfect matching problem which was shown to be polynomially solvable by Edmonds in 1965 [71].

Note that since k is unrestricted in problem 5, problem 4 is a special case of problem 5 and hence the latter is NP-complete also. However, if k is restricted to a *fixed* value, we have yet another problem, this one introduced by Papadimitriou and Yannakakis [72]. (See also [73].)

6. Let the positive integer k have a *fixed* value (say, 10, for example). Given a bipartite graph G with its edges colored in k colors and a set of non-negative integers $c_1, \ldots, c_k$, does G contain a perfect matching F such that F contains $\leq c_i$ edges of color i, for all i?

The difference between problem 6 and problem 5 is important to see, although it is somewhat subtle. In problem 6, the number of edge classes (i.e., "colors") is fixed or bounded; in problem 5, k is not bounded, but may assume values as large as one likes as part of the input string to the problem.

Papadimitriou and Yannakakis [72] showed problem 6 to be polynomially equivalent to five other problems, including the next problem in our list—EXACT MATCHING.

7. Given a graph G and a set of distinguished edges $R \subseteq E(G)$ (call these edges "red") and an integer $k > 0$, is there a perfect matching of G which contains *exactly* k red edges?

Note that EXACT MATCHING can also be thought of as the special case when $k = 2$ and $R_1 \cap R_2 = \emptyset$ of the RESTRICTED MATCHING problem.

Although the complexity of problems 6 and 7 remains unknown, even for bipartite graphs, Barahona and Pulleyblank [74] have shown EXACT MATCHING to be polynomially solvable when G is *Pfaffian*. The class of Pfaffian graphs is discussed in [1] and we will meet them again in Section 7 of the present paper. Suffice it to say here that they include all $K_{3,3}$-free graphs and hence all *planar* graphs as well. More recently, V.V. Vazirani [75,76] has shown EXACT MATCHING to be in parallel class NC for $K_{3,3}$-free graphs. We shall return to EXACT MATCHING in Section 5.

Given a matching M in a graph G, C is an **alternating cycle** with respect to M if C is a (necessarily even) cycle in which every second edge belongs to M. Matching M is **alternating cycle free** if no such C exists. To test if a graph G has an alternating cycle free perfect matching can be done in polynomial time, for clearly G has a cycle free perfect matching if and only if G has a *unique* perfect matching. To test for the latter property, let $M = \{e_1, \ldots, e_{n/2}\}$ be any perfect matching for G and test $G - e_i$ for each i in turn to see if it contains a perfect matching, using the polynomial algorithm of Edmonds or any

of its variants.

On the other hand, consider the following two related problems.

8. Does a *bipartite* graph G have a perfect matching which has no alternating 4-cycle?

9. Given a bipartite graph G and an integer $k > 0$, does G have an alternating cycle free matching of size at least k?

Pulleyblank [77] has proved both these problems NP-complete by reducing 3SAT to each. This work was done in connection with minimizing setups in precedence constrained scheduling.

Varying our demands somewhat yet again, let us call a matching M in graph G **induced** if no two edges of M are joined by a edge in $E(G) - M$. Consider now the *induced matching problem.*

10. Given graph G and an integer $k > 0$, is there an induced matching in G of size $\geq k$?

This problem has been proved NP-complete even for *bipartite* graphs by Stockmeyer and V.V. Vazirani [78] and later, independently, by Cameron [79]. (The first two authors actually showed more. Let the distance between two edges e and e' be the length of a shortest path from an endvertex of e to an endvertex of e'. A **δ-separated matching** M is a matching in which the distance between any two edges is at least δ. Stockmeyer and Vazirani showed that for each $\delta \geq 2$ the problem "Given a graph G and an integer $k > 0$, does G have a δ-separated matching of size at least k?" to be NP-complete, even for bipartite graphs regular of degree 4.)

On the other hand, Cameron showed maximum induced matching (= maximum 2-separated matching) is polynomial—in fact in NC—for all *chordal* graphs. (A graph is **chordal** if every cycle of length at least 4 has a chord, that is, a edge not in the cycle, but which joins two vertices of the cycle.)

In the Stockmeyer and Vazirani paper, two other interesting variations are shown to be NP-complete. These are *maximum TR-matching* and *maximum star matching.* Both are relevant to network testing of various sorts.

A **TR-matching** in graph G is a pair (M, λ) where M is a matching and λ is a labeling function which assigns to each vertex of G one of the three labels from $\{T, R, \Lambda\}$. Here "T" stands for "transmitter", "R" for "receiver" and "Λ" for "neither". In addition, the labeling λ is subject to the following conditions:

(a) $\lambda(v) = \Lambda$ whenever M does not cover vertex v,

(b) if edge $uv \in M$, then precisely one of u and v has label T, the other R, and

(c) if $uv \in E(G) - M$, then $\{\lambda(u), \lambda(v)\} \neq \{R, T\}$.

Condition (c) says that no transmitter is connected to a receiver other than the one to which it is matched by M. The **size** of a TR matching is the cardinality of M.

11. Given graph G and integer $k > 0$, is there a TR-matching in G of size at least k?

The motivation for this problem is reasonably clear. We want to test the network by sending a signal simultaneously from all transmitters (T's) to their receivers (R's). Condition (c) precludes the possibility of "jamming"; that is, receivers receiving test signals from two different transmitters.

Star-matching arises from a different testing procedure. A **star-matching** is simply a labeling function λ from the set of vertices $V(G)$ into the set $\{T, R\}$ such that for each vertex u with $\lambda(u) = R$ there exists another vertex v adjacent to u such that $\lambda(v) = T$. The **size** of a star-matching is the size of the set of vertices having R labels.

12. Given graph G and integer $k > 0$, does G have a star-matching of size at least k?

Problems 11 and 12 were shown to be NP-complete for *bipartite* graphs by Even, Goldreich and Tong [80]. Stockmeyer and Vazirani show them to be NP-complete for *all cubic* graphs.

Returning to induced matching for a moment, we point out that Stockmeyer and Vazirani refer to it as "risk-free marriage"! (A moment's reflection on the part of the reader will undoubtedly reveal why.) This leads us to another matching variation which has been widely studied over the past twenty years or so: *The Stable Marriage Problem.*

Fortunately, for this problem we have a quite recent and comprehensive survey in the form of the book of Gusfield and Irving [81].

As in the case with matching in general, studies here naturely divide into the treatment of the bipartite case (the *Stable Marriage Problem*) and the general—i.e., not necessarily bipartite—case (the *Stable Roommates Problem*).

We treat the bipartite version first. Suppose we have a bipartite graph G with vertex bipartition $V(G) = A \cup B$ and suppose that $|A| = |B|$. Each man (member of A) and each woman (member of B) has a complete list of preferences for the member of the set of the opposite sex which is a strict order relation. A matching M is said to be **unstable** if there exists a man and woman who are not matched by M, but *each* prefers the other over his/her partner in the matching M. A matching is said to be **stable** if it is not unstable. The problem then is:

13. Given a bipartite graph as above with ranked preferences, does G have a perfect matching which is stable?

Of course as usual, there are really *two* fundamental related problems here. First, does a stable matching exist and second, if so, can it be found efficiently. The answer, fortunately, is "yes" to both questions.

In 1962, Gale and Shapley [82] showed that a stable matching *always* exists and provided an $O(n^2)$ algorithm to find one.

The more general problem of Stable Roommates is defined in a manner similar to the Stable Marriage Problem, except that the underlying graph G need not be bipartite (although it should have an even number of vertices) and each vertex (= student) has a strict ordering of all other vertices (= possible roommates).

14. Given a graph G on an even number of vertices with preference lists for each vertex, does G have a stable matching?

The fundamental difference between the Marriage and Roommates problems turns out to be that the latter *need not* have a stable matching.

The complexity of the Stable Roommates Problem was settled in 1985 by Irving [83] who found an $O(n^2)$ algorithm which either outputs a stable matching or indicates that

none exists.

These two problems are unique in this paper in the sense that it is known that the algorithms are *asymptotically optimal.* That is, it has been proved that *any* algorithm for finding a stable matching (even in the bipartite case) must require at least cn^2 time, for some constant $c > 0$. This was proved by Ng and Hirschberg in 1988 [84].

If one relaxes the demand on *strict* preference lists to allow the possibility of ties, one must agree on a suitable redefining of stability. However, for two of the "most natural" such definitions (called *super-stable* and *strongly stable* by Gusfield and Irving), the Gale-Shapley algorithm can be extended to produce a stable matching (or report the existence of none) in polynomial time in the *bipartite* case.

However, for the non-bipartite Roommates Problem, if ties are allowed, the problem becomes NP-complete. This was proved by Ronn [85,86].

There are a host of variations on Stable Marriage and Stable Roommates, some in P, some NP-complete and some the complexity of which is unknown. (See the Gusfield and Irving book [81].)

We will finish our treatment with one more such variation. Note that we now return to the original requirement that the preferences be *strict.* For a roommates matching M, let the **value of** M be

$$v(M) = \sum_{uv \in M} (r(u,v) + r(v,u))$$

where $r(u,v)$ denotes the ranking of v by u. The *Optimal Roommates Problem* is:

15. Given a graph G, compute the minimum value of $v(M)$ taken over all stable matchings M.

Feder [87] has recently proved that this problem is NP-complete, but the bipartite (or Marriage) version is, in fact, polynomial.

We conclude by reporting that the problem of *counting* stable matchings, even in the bipartite case, is #P-complete [86].

Now we veer in a different direction. Maximum matching can be thought of as a "packing problem" in which one wants to find the largest number of edges which are mutually vertex-disjoint. Suppose now we attempt to generalize the notion of "edge" in the packing to another type of subgraph Let us call the following the *H-matching problem* and denote it by **HMATCH**.

Given graphs G and H is there a spanning subgraph of G consisting of vertex-disjoint copies of H?

Of course if $H = K_2$, we just have the perfect matching problem and the solution is polynomial. But if H has any component having 3 or more vertices, Kirkpatrick and Hell [88,89] proved that the problem is NP-complete. (See also [90,91].) In all other cases, the problem is polynomial.

We can modify this problem in yet another way. Instead of *one* fixed graph H, as our subgraph to be packed, let us allow a choice of graphs from a certain specified family. Let $\mathcal{H} = \{H_1, H_2, \ldots,\}$ be a finite or infinite family of graphs. An **$\mathcal{H}$-factor** for G is then a

vertex-disjoint collection of subgraphs of G which together cover $V(G)$ and each member of the collection comes from $\mathcal{H}$. For example, if $\mathcal{H} = \{C_3, C_4, \ldots\}$, the family of all cycles, then an $\mathcal{H}$-factor for G just becomes a **2-factor**.

Hell and Kirkpatrick study a variety of possibilities for $\mathcal{H}$. For example, if $\mathcal{H}_1$ is the set of all cycles just mentioned, the problem is polynomial. If $\mathcal{H}_2 = \{K_2\} \cup \mathcal{H}_1$, the packing sought is called a **perfect 2-matching** and again the problem is polynomial.

If $\mathcal{H}_3 = \{K_{1,n} | n \geq 3\}$ (i.e., the family of "stars"), the problem is NP-complete. On the other hand, if we add to $\mathcal{H}_3$ either K_1 or K_2 (or both), and call the resulting class $\mathcal{H}_4$, the problem clearly becomes polynomial. A close relative to this problem is obtained as follows. Let $\mathcal{H}_5$ be any subset of $\{K_{1,n} | n \geq 1\}$ with the property that for some t, $K_{1,t} \notin \mathcal{H}_5$, but $K_{1,t+1} \in \mathcal{H}_5$. Then the $\mathcal{H}_5$ problem is NP-complete.

One final example is obtained by letting $\mathcal{H}_6$ be any family of complete graphs. Then the $\mathcal{H}_6$ packing problem is polynomial if K_1 or K_2 is in $\mathcal{H}_6$ and NP-complete in all other cases. Proofs of all these results, as well as other related results, may be found in [92,93,90,94,95,91,96,97].

More recently, the H-matching problem has been investigated for *planar* graphs [98]. (Here again, let us note, H is a single fixed graph.) If $H = K_3$ or $K_{1,3}$ (the "claw"), Dyer and Frieze [99] showed HMATCH to be NP-complete in the plane. Even more recently, Berman et al. [98] have shown that if H has at least 3 vertices, then *maximum* planar matching version of HMATCH is NP-complete. Surprisingly, *perfect* planar matching version of HMATCH is another story! If H has at least 3 vertices and is connected and *outerplanar*, they show that the problem is NP-complete. On the other hand, if H is a triangulated graph with at least 4 vertices, there is an $O(n)$ algorithm for the problem.

A characterization of those H's for which perfect planar HMATCH is polynomial remains unknown. An approximation algorithm for planar H-matching is obtained in [100].

Now let us look at a quite different problem associated with matching. Edmonds showed that that matching can be formulated as a linear program. There now exist several polynomial LP algorithms such as the ellipsoid method and Karmarkar's method. The problem with using these directly is that there exist an exponentially large number of inequalities which must be furnished in order to formulate the matching problem as an LP.

However, the ellipsoid method has the property that if the inequalities can be fed to it as needed, it can solve the LP in polynomial time. Padberg and Rao [101] described an algorithm to do this for the matching polyhedron. Thus their algorithm, when used in conjunction with the ellipsoid method, provides another method for solving matching problems in polynomial time.

But can one somehow solve the matching problem in polynomial time using a more standard LP algorithm like the aforementioned simplex method or Karmarkar's algorithm? This remains an open question.

Barahona [102] has made some progress on this question by showing that one can solve matching via a *polynomial number* of LP's, each *polynomial in size*. He has also shown that in the case when G is *planar*, this polynomial number of polynomial size LP's can indeed be reduced to only one [103]. It should be mentioned in this connection that a

planar graph can indeed have an exponential number of facets in its matching polytope. (Cf. Gamble [104].)

On the negative side, however, Yannakakis [105] has shown that no *symmetric* LP formulation of polynomial size for matching on the complete graph K_{2n} is possible. (A formulation of the matching problem is **symmetric** if any extra variables and the roles they play are independent (up to permutation) of the order in which the graph is examined.)

To close this section of our paper, we make a fleeting visit to the land of matroids. For details the reader may refer to [1] or [106], or to many other reference sources for matroid theory. Suppose $\mathcal{M} = (E, I)$ is a matroid where E is the ground set and I is the family of independent subsets of E. Suppose F is a pairing of all the elements of the ground set E. A set $A \subseteq E$ is a **parity set** if for every element $e \in A$, its mate under F is also in A. Then the MATROID PARITY problem is:

Given $\mathcal{M} = (E, I)$, a pairing F and an integer $k > 0$, is there a parity set A in $\mathcal{M}$ of size at least k?

MATROID PARITY is provably exponential and this result does not depend upon the assumption that P $\neq$ NP! This was shown by Lovász in 1978 [107]. If the matroid is *linear*, however, he gave a polynomial algorithm, albeit relatively slow. More recently, faster algorithms have been found [108].

4. Vertex Packing Variations and their Complexity

As mentioned earlier in this paper, the vertex cover problem is one of the original NP-complete problems in the list of Karp [61]. Later it was proved NP-complete even when restricted to cubic planar graphs [21], to triangle-free graphs [109] and several other families listed in [21,110]. Although vertex packing is polynomial for $K_{1,3}$-free graphs (see Minty [19] and Sbihi [20] mentioned earlier), it is still NP-complete for $K_{1,4}$-free graphs [19]! (See also [111].) For even more classes for which vertex cover is NP-complete, see also Mahadev [112].

Recall that earlier, along with claw-free graphs, we also mentioned line graphs and bipartite graphs as classes for which VP is polynomial. There are several other classes for which polynomiality for vertex packing has been shown, but too recently to appear in the Garey and Johnson book.

The most famous of these is surely the family of *perfect* graphs. Let $\omega(G)$ denote the size of any largest complete subgraph in G. This is called the **clique number** of G. Clearly, $\omega(G) \leq \chi(G)$ for any graph G, where $\chi(G)$ denotes the **chromatic number** of G. Graph G is said to be **perfect** if $\omega(G') = \chi(G')$ for every induced subgraph G' of G. The concept of perfection is due to Berge [113] in the early 1960's and the class of perfect graphs is now known to include many other well-known families such as bipartite graphs, line graphs of bipartite graphs, interval graphs, comparability graphs and triangulated graphs—as well as the complements of all such graphs. In fact, Lovász, in a celebrated 1972 result [114], proved that for *any* perfect graph the complement must also be perfect. (For general references on perfect graphs, we recommend [115], [116] as well as [117,118,119].)

It is a highly non-trivial fact, proved by Grötschel, Lovász and Schrijver, that vertex packing is polynomial for the class of all perfect graphs. The proof uses the *ellipsoid method*, the first polynomial LP algorithm to be discovered. For a thorough account of this see [117,118,120,121,122]. In fact, they prove polynomiality for a larger class, namely the so-called *h-perfect* graphs. The *h*-perfect graphs are defined as those graphs the stable set polytopes of which are defined by certain families of linear inequalities—in this case the so-called non-negativity constraints, clique constraints and odd cycle constraints. Unfortunately, no purely graph-theoretical characterization of *h*-perfect graphs is yet known.

One last remark about perfect graphs is in order. Although the complexity of showing a graph to be perfect is not known, showing imperfection is in class NP. (According to Berge and Chvátal [116], this result is attributable to Edmonds and Cameron.) For additional work along these lines, see references [123,124].

Now let us return to bipartite graphs and recall the fundamental result of Kõnig which says that $\nu(G) = \tau(G)$ where $\nu(G)$ is the size of any maximum matching and $\tau(G)$ is the size of any minimum vertex cover [3,4]. This is an archetypal example of a "minimax" theorem in graph theory. Such theorems have grown in importance since the discoveries of the last twenty years or so indicating that graph theory and linear programming can profitably be brought together. (See [1] for just one of many references on this subject. See also [2].)

It is important to realize that the above minimax equation holds for some non-bipartite graphs as well; for example, consider the 4-vertex graph obtained by attaching a pendant edge to a triangle. Graphs satisfying the minimax equation are said to have the **König Property**. First of all, there are several characterizations of these graphs which lead to polynomial recognition algorithms for the graphs in the class. (See [125,126,127,1].) Trivially, these graphs have polynomial algorithms to find the *size* of a maximum independent set via the matching algorithm. However, none of the above references explicitly gives a polynomial algorithm for *finding* a maximum independent set—i.e., the search problem. However, in [127], there is a polynomial algorithm for the search problem implicit in Lemma 3.3. This result depends upon the so-called Gallai-Edmonds decomposition of a graph, a canonical decomposition of graphs in terms of their maximum matchings. This decomposition can be found in polynomial time via Edmonds' algorithm. The details may be found in [1]. Also see the thesis of Korach [128].

It turns out, however, that a polynomial algorithm for finding a maximum independent set in a graph with the Kõnig Property has been around quite a bit longer than the method referred to in [1]. Define a **2-cover** of a graph G to be an assignment of weights 0, 1 and 2 to the vertices of G such that the sum of weights of the two endvertices of any edge is at least 2. The sum of all weights is called the **size** of the 2-cover. The minimum size of any 2-cover of G is denoted by $\tau_2(G)$. It can be shown (see Corollary 6.3.4 of [1]) that a graph G has the Kõnig Property if and only if it satisfies $\tau_2(G) = 2\tau(G)$. In a 1975 paper, Nemhauser and Trotter [129] gave a polynomial algorithm which, when applied to an arbitrary graph, either produces a maximum independent set or shows that $\tau_2(G) < 2\tau(G)$ and thus that the graph does not have the *not* Kõnig Property. (This was before the name "Kõnig Property" had yet been coined, however.) For further discussion on this approach, the reader is referred to [130] and [131].

Another class of graphs with polynomial algorithms for VP are the *well-covered* graphs mentioned earlier. Recall that a graph is well-covered if every maximal independent set is in fact maximum. In other words, these are the graphs for which the greedy algorithm for a maximal independent set *always* results in a maximum independent set. The polynomiality for VP for these graphs is thus trivial. What is *not* trivial about well-covered graphs is recognizing them!

A moment's reflection will tell the reader that well-covered graphs are in co-NP. In fact, very recently it has been shown that in fact well-covered graph recognition is co-NP-complete. (See [132,133].) But membership in NP is unsettled. In fact, it is thought by most that membership in NP is highly unlikely, for such a result would imply that NP = co-NP, a situation considered almost as unlikely as P = NP!

The concept of a well-covered graph was introduced in [57]. In the past five years or so, there has been a surge of interest in this graph class and the interested reader may want to consult the forthcoming survey [134]. See also [135,133,132].

Next we would like to briefly discuss the so-called *α-critical* graphs. Let the *independence number* of graph G (i.e., the size of any largest independent set in G) be denoted by $\alpha(G)$. A graph G is said to be **α-critical** if $\alpha(G-e) > \alpha(G)$, for all edges $e \in E(G)$. It may well be that the recognition problem for these graphs belongs to neither NP nor to co-NP! At this point, no one knows.

If these graphs *were* in NP, then one could give a "good characterization" of $\alpha(G)$ for any graph. (In other words, one could show that determining $\alpha(G)$ belongs to NP$\cap$co − NP [123].)

There has been quite a lot of interest in obtaining structural properties of this family of graphs. For an overview of the structural results presently known for these graphs, see [1].

In 1965, Hajnal [136] proved that in any α-critical graph G, maxdeg $v \leq |V(G)| - 2\alpha(G) + 1$. Denote the quantity $|V(G)| - 2\alpha(G)$ by $\delta(G)$. Gallai suggested that studying connected α-critical graphs G by means of the value of $\delta(G)$ might prove profitable. The parameter $\delta(G)$ has come to be called the **Gallai class number** of G for this reason. Clearly the only connected α-critical graph with $\delta = 0$ is K_2. It is also easy to see that the only connected α-critical graphs having $\delta(G) = 1$ are the odd cycles. From this point on, the situation rapidly becomes more complex.

Andrásfai [137] showed that the only connected α-critical graphs with $\alpha(G) = 2$ are the even subdivisions of K_4. A deep result due to Lovász some eleven years later [121] showed that for *any* value of δ, the class of α-critical graphs G having $\delta = \delta(G)$ must arise from a *finite* class of "basis" graphs via even subdivisions. For $\delta = 3$, it has been shown that there are precisely four basis graphs, while for $\delta \geq 4$, it is known only that the number of basis graphs is bounded above by a rather complicated function exponential in δ.

Returning to Andrásfai's result for a moment, let us compare it with a result of Chvátal [138] which says that every α-critical graph G with $\delta(G) \geq 2$ must contain a subdivision of K_4. Moreover, Chvátal also showed that any graph which does *not* contain a subdivision of K_4 (i.e., a so-called *series-parallel* graph), has a polynomial algorithm for VP.

Chvátal also conjectured that all α-critical graphs with $\delta \geq 2$ must in fact contain an

even subdivision of K_4. This conjecture was settled in the affirmative by Sewell [139,140] and Sewell and Trotter [141] in 1990. Moreover, in [139,140], Sewell also showed that if graph G does not contain an *even* subdivision of K_4, then the VP problem for G is polynomial.

Thus we have a kind of "separation" result in that we have a class of graphs which are *not* α-critical, but which have a polynomial VP algorithm. It is interesting that the polynomial VP algorithm of Sewell uses the ellipsoid method also. It is an open question as to whether use of the ellipsoid method can be avoided in this case.

One final remark is in order regarding α-critical graphs. We stated earlier that it may well be that recognizing these graphs is neither in NP nor in co-NP. What then, if anything, can be said about the complexity of this problem?

In 1982, Papadimitriou and Yannakakis [142], in an attempt to classify the complexity of facets of the polytope which arises in the LP formulation of the traveling salesman problem, defined a new complexity class D^P. (See also [73,143,50].) In terms of languages (i.e., sets of binary strings), D^P is defined as the set of all languages $L_1 \cap L_2$ where L_1 is in NP and L_2 is in co-NP. In terms of problems, we perhaps can best illustrate this class with the example of EXACT VERTEX PACKING:

Given a graph G and an integer $k > 0$, is $\alpha(G) = k$?

This can be thought of as the intersection of two problems. The first is our old friend VP which says "Given G and $k > 0$ does G have an independent set of size at least k?". This is in NP as we have seen earlier. As the second problem, consider: "Given G and $k > 0$, does G not have an independent set of size at least $k+1$?". In the second problem, if the answer is "no", it is easily certifiable by giving an independent set which has size at least $k+1$.

It follows by definition that $NP \cup co-NP \subseteq D^P$. It should be emphasized that there is an important distinction between a class defined in terms of *two* problems, one in NP, the other in co-NP and a class defined in terms of a single problem simultaneously belonging to NP and to co-NP. D^P is an example of the former; $NP \cap co-NP$ is an example of the latter.

It turns out that not only does the problem of determining the facets of the VP polytope lie in this new class, but so does EXACT VERTEX PACKING. In fact, both are D^P-complete! (See [73,142].) Somewhat later in 1985, Papadimitriou and Wolfe [143] announced that V.V. Vazirani has shown that the recognition problem for α-critical graphs is also D^P-complete, although we are unaware of a published proof to date.

We conclude this section with one brief remark on *approximating* $\alpha(G)$. Some time ago, Garey and Johnson [21] proved that in a sense this is a hopeless task. More specifically, they showed that if one could find a polynomial time algorithm which, given an $\epsilon > 0$, outputs an approximate value $\alpha^*(G)$ in the sense that $|\alpha(G) - \alpha^*(G)| < \epsilon$, then in fact there must be a polynomial algorithm for MAXMVP and hence P = NP!

There are, of course, other measures of "approximation". For example, suppose $\alpha^*(G)$ is an estimate for $\alpha(G)$. One could consider the *ratio* $\alpha(G)/\alpha^*(G)$ as a measure of "goodness" of approximation. But for this case as well, the news is not good. There is no known algorithm for FINDVP which guarantees a ratio any better than $O(n^\epsilon)$, where as usual n

is the input size and $\epsilon > 0$.

Some recent results by Feige, Lovász, Goldwasser, Safra and Szegedy [144,145,146] are of special interest here. These authors show that if one could approximate $\alpha(G)$ in polynomial time to within a factor of $2^{(\log n)^{1-\epsilon}}$, then NP $\subseteq$ QP, where QP denotes *quasi-polynomial time*; that is, $O(2^{\log^c n})$. This set inclusion would then imply NEXPTIME = EXPTIME. (This equality would also be implied by P = NP.) It is also shown that if one can approximate the independence number within a *constant* factor in polynomial time, then in fact every NP problem can be solved in $n^{O(\log \log n)}$ time.

The proofs employ techniques of interactive proof systems. (See the "IP" at the top of our complexity class diagram.)

But lest we leave the reader on too negative a note, it has been shown by Boppana and Halldórsson [147] that there is a *polynomial* time algorithm which will approximate $\alpha(G)$ within a factor of $n/\log^2 n$.

In view of the difficulties encountered in approximating the size of a maximum independent set, one may find the following surprising, due to the complementary connection between independent sets and vertex covers. Gavril [21] first observed in 1974 that there is a straightforward polynomial approximation algorithm which for any graph G supplies a number $\tau^*(G)$ such that $\tau^*(G)/\tau(G) \leq 2$! Remember that by Gallai's result [5], $\tau(G) + \alpha(G) = |V(G)|$. Now simply construct any maximal matching M, say of size k. Then the set of endvertices C of M is a set of $2k$ vertices and by the maximality of M, C is a vertex cover. Now every cover of G must in particular cover M, so $\tau(G) \geq k$. Thus we have $2\tau(G) \geq 2k = |C|$ and hence $|C|/\tau(G) \leq 2$ as claimed.

This ratio of 2 was improved to a factor of $2 - \Omega((\log \log n)/\log n)$ by Bar-Yehuda and Even in 1983 [148] and independently by Monien and Speckenmeyer in 1985 [149].

5. Matching in Parallel

It is a good starting point for this section to recall that without any of the various additional "bell and whistle" conditions discussed in Section 3, the problem of finding a maximum matching for general graphs can be done in polynomial time via Edmonds algorithm or any of its descendants. Of course these algorithms are all *sequential*. Note also that (a) finding a perfect matching (or showing none exists) and (b) finding a maxim*al* matching are both sequentially polynomial as well. Polynomiality of (a) follows from Edmonds algorithm and polynomiality of (b) is essentially trivial by the greedy algorithm.

The situation is quite different when one moves to a parallel setting. It is not at all clear how to efficiently parallelize even greedy matching, let alone Edmonds' algorithm. In fact, perhaps the outstanding open question regarding matching in parallel is whether or not maximum matching (or perfect matching, for that matter) belongs to NC. (There are certainly parallel algorithms for maximum matching, however. See [150,151,152,153,154]. They simply are not *polylog* time algorithms.)

But let us begin with the (apparently) even simpler problem of finding a maxim*al* matching. Let us denote this problem by MAXLMATCH. So that the reader won't be kept in suspense, let us announce that MAXLMATCH is now known to be in NC (in fact, in NC^2). But just how it got there is an interesting story.

In 1980, Lev [155] showed that for *bipartite* graphs, MAXLMATCH was in NC^4. Actually, she gave three different algorithms for this problem. In addition—and this seems not to be widely realized—she also gave an NC^4 algorithm for finding a maximum matching (henceforth MAXMMATCH) in any *regular bipartite* graph by means of an NC^4 edge coloring algorithm for this family. The bound was improved to NC^2 in [156]. The complexity for general regular graphs remains open, although see [157].

No processor count was mentioned by Lev, although Israeli and Shiloach [158] assess her processor bound as $O(m^3/\log m)$.

Four years later, Karp and Wigderson [159,160] extended the result of Lev by showing MAXLMATCH to be in NC^4 for *all* graphs. Their algorithm employed $m+n$ processors. The Karp–Wigderson algorithm was actually a special case of the first NC maximal independent set (hereafter MAXLVP) algorithm. We shall return to their algorithm in the next section.

In [158], Israeli and Shiloach give another NC^4 algorithm for MAXLMATCH using $m+n$ processors which implemented on a CRCW-PRAM reduces to NC^3. Shortly thereafter, Israeli and Itai [161] found a *randomized* algorithm for MAXLMATCH which lowers the polylog time exponent by 2, thus placing the algorithm in RNC^1. (The reader should note that this is the first use of a randomized algorithm in this paper, but not the last!)

A side remark is in order at this point. Luby [162,163] proved MAXMMATCH to be in NC^5 with $O(n)$ processors. Although this bound is not as good as that of Israeli and Shiloach, it deserves mention because it introduced a new and sophisticated technique the full potential of which has probably not yet been realized. The procedure we refer to formulates a randomized algorithm (in this application to both MAXLMATCH and to MAXMLVP) and then removes the randomness to obtain a deterministic algorithm. Such techniques were used in Karp and Wigderson [159,160] and previously by Luby himself [164,165] primarily in connection with MAXLVP. (See the next section.) But these older procedures have the unpleasant side effect of producing a rather large blow up in the number of processors required. In [162,163], Luby develops a new approach making clever use of a new probabilistic space which ultimately permits removal of randomization with *no* increase in the number of processors required.

However, it was in [164,165] that Luby reduced the complexity of the problem to NC^2, again proceeding as did Karp and Wigderson to obtain his bound as a special case of an algorithm for MAXLVP. Again, we will have more to say about this in the next section.

Now let us turn to parallel algorithms for perfect and maximum matching. To further muddy the waters, in the parallel case, we shall have to differentiate between the problems of *deciding* if a perfect matching exists (hereafter **?PM**) and the *search* version of the problem (FINDPM). Remember: none of these three problems is known to be in NC.

There are some intriguing new questions which arise in parallel computing which are not really significant in the sequential situation. One of these asks: "What is the difference in complexity between decision problems and search problems?" The interested reader may consult [166] for a diversion into the land of rank and independence oracles and their relative power.

In 1979, Lovász [167] proposed a test for ?PM by giving a randomized procedure for testing if a certain matrix is non-singular. Borodin, von zur Gathen and Hopcroft [168,169]

then combined this with an NC^2 algorithm already developed by Csánky [170] for testing matrix non-singularity to give an RNC^2 algorithm for ?PM.

For the search problem FINDPM, a breakthrough was obtained in 1985 by Karp, Upfal and Wigderson [171,172] who found the first RNC algorithm. Their procedure actually placed the problem in RNC^3. Shortly thereafter, Galil and Pan [173,174] considerably improved the processor bound. In 1987, Mulmuley, Vazirani and Vazirani [175,176] put the problem in RNC^2 with a faster algorithm with processor bound $O(n^{3.5}m)$.

Now what about MAXMMATCH? Actually, Mulmuley, Vazirani and Vazirani [175, 176] showed that this problem is also in RNC^3 as well as the following related search problems:

(a) Find a maximum *weight* perfect matching in a graph the edge weights of which are given in *unary*, and

(b) Find a matching covering a set of vertices of maximum weight in a graph where the weights on the vertices are given in *binary*.

Moreover, they show that if any of these three search problems is in NC, they all are. (The Galil–Pan improved processor bounds apply here too.)

A clever observation by Karloff [177] can be used to turn all these algorithms of Karp, Upfal and Wigderson and Mulmuley, Vazirani and Vazirani from Monte Carlo to Las Vegas, so all are now known to actually lie in "zero-error" class ZNC^2.

It is interesting that a close relative to the above three problems has unknown complexity:

(c) Find a *minimum weight perfect matching* in a graph with edge weights given in *binary*.

Before leaving these two papers, we should also inform the reader that Mulmuley, Vazirani and Vazirani were also able to show that EXACTMATCH lies in RNC^2 as well. Recall from Section 3 that it is not known if this problem can be solved deterministically in polynomial time.

Much of the information given in this section down to this point can be found (and in considerably greater detail) in the two excellent surveys by Galil [178,179] which appeared nearly simultaneously in 1986.

We now return to the basic unsolved problems motivating this section; that is: Are any of ?PM, FINDPM or MAXMMATCH in NC?

Although these problems remain open in general, membership has been affirmed for various graph classes.

In addition to regular bipartite graphs, as discussed earlier, (see [155,156]), FINDPM has been shown to be in NC for the following graph families:

1. Claw-free graphs (and hence for line graphs as well). Chrobak and Naor [180] give an NC^2 algorithm. See also Naor [181].

2. Planar bipartite graphs. This was done by Miller and Naor in 1989 [182]. Note that FINDPM remains open for planar graphs in general. Here is an instance where, on the other hand, ?PM *is* known to be in NC! This is an immediate corollary of the fact that counting perfect matchings (hencforth #PM) is in NC^2 when the graph is planar. We shall deal with this again in Section 7 below.

3. Dense graphs. A graph is said to be **dense** if mindeg $G \geq n/2$. Dahlhaus and Karpinski [183] found an NC^2 algorithm requiring $O(n^8)$ processors and somewhat later, Dahlhaus, Hajnal and Karpinski [184,185] found an NC^4 procedure using a linear number of processors.

Note that a dense graph (on an even number of vertices) *always* has a perfect matching. This is an immediate corollary to the classical theorem of Dirac which says more, namely that such a graph must have a Hamilton cycle.

A very interesting "negative" result is also proved in these three papers. Namely, if one drops the degree bound to $n/2 - \epsilon$ for any $\epsilon > 0$, then the problem is just as hard (under NC-reductions) as FINDPM for *general* graphs!

4. Strongly chordal graphs. A graph G is **chordal** if every cycle of length > 3 has a chord. This is equivalent to having a *perfect elimination scheme* $<$ on the vertex set; namely, if $(u,v),(u,w) \in E(G)$, then $(v,w) \in E(G)$. **Strongly chordal** graphs can be defined by imposing the additional requirement that the perfect elimination scheme also satisfies: if $x < u$ and $y < v$ and if $(x,y),(x,v),(y,u) \in E(G)$, then $(u,v) \in E(G)$.

Strongly chordal graphs are in NC^2 by Dahlhaus and Karpinski [186]. Although FINDPM is in NC^2 for strongly chordal graphs, for the wider class of chordal graphs, it is as hard as the general bipartite case, the parallel complexity of which, as we have remarked several times already, is unknown.

5. Co-comparability graphs. A **precedence graph** is an acyclic transitively closed directed graph. A **comparability graph** is an undirected graph which can be oriented so as to become a precedence graph. **Co-comparability graphs** are the complements of comparability graphs. Helmbold and Mayr [187,188] have produced an NC^2 algorithm for these graphs.

There is a close relationship between co-comparability graphs and the so-called *two processor scheduling problem*. As corollaries, one obtains here NC^2 algorithms for FINDPM for the class of permutation graphs and partial orders of dimension 2, as well as interval graphs.

An NC algorithm for co-comparability graphs was independently found by Kozen, Vazirani and Vazirani [189], although no polylog time exponent is given. This paper is also interesting from another point of view. Although the membership of ?PM in NC is a famous unsolved problem, here these three authors give an NC algorithm for the closely related problem: "Does graph G have a *unique* perfect matching?" Moreover, Rabin and Vazirani [190] give an NC^2 algorithm for FINDPM, in the case when G has a unique perfect matching.

6. Bipartite graphs with a polynomially bounded number of perfect matchings. Grigoriev and Karpinski [191] give an NC^3 algorithm for this class. In fact, the authors do more. For such graphs, they show that finding *all* perfect matchings is in NC^3. Moreover, if $\Phi(G)$ is bounded by a *constant*, they find all perfect matchings with a faster—NC^2—algorithm.

An interesting sidelight here is the problem of recognizing such graphs! This is leading us perilously close to yet another important problem: *counting* the number $\Phi(G)$ of perfect matchings in a graph G. This has been shown to be #P-complete, so hope for a polynomial algorithm, much less an NC algorithm, to compute $\Phi(G)$ is dim indeed. However, Grigoriev

and Karpinski do give an RNC^3 algorithm which, given any polynomial cn^k, will decide if $\Phi(G) \leq cn^k$.

For the problem of ?PM in bipartite graphs with a polynomially bounded number of perfect matchings, these two authors also give an NC^2 algorithm in their paper. They also state that their results can be extended from bipartite graphs to graphs in general, but no proofs are given.

Let us insert at this point a few remarks on the problem ?PM. In the paper of Grigoriev and Karpinski discussed in the preceding three paragraphs, one also finds the result that ?PM belongs to NC^2 for graphs with a polynomially bounded number of perfect matchings. We will see in the last section of our paper that exact counting of perfect matchings—#PM—is in NC for certain classes of graphs; e.g., planar graphs. Thus by default, ?PM is in NC for all such classes. (Recall that #PM is #P-complete and therefore NP-hard in general.)

Now let us return to the maximal matching problem—MAXLMATCH—discussed at the beginning of this section. As we noted above, it is now known to be in NC, although not without a struggle! Motivated still by the apparent difficulty of doing greedy procedures in parallel, researchers have posed and investigated the folowing variant of MAXLMATCH.

Suppose the edges of a graph G are numbered 1 through m. The *lexicographically first maximal matching problem*—LFMAXLMATCH—is stated as follows. Does the lexicographically first maximal matching in G contain edge m? Is the problem in NC or perhaps even P-complete? These are open questions. (See [190,192,193,194].)

In the latter three references, the problem is one of several used to motivate the creation of a new complexity class—**CC**. (This will be the last class in our lattice diagram to be discussed.) This class is discussed at length in [193,192,194] with perhaps the most succinct treatment in [22].

First we must define the *circuit value problem*—or **CV**: "Given a Boolean circuit with n inputs and a single output, together with a binary input value for each input gate, is the output = 1?"

CV is known to be P-complete under logspace reductions. (See [195].)

Now we modify the CV problem as follows. Suppose we restrict all circuit elements to have precisely two inputs and two outputs, one output yielding $x \wedge y$, the other yielding $x \vee y$ for binary inputs x and y. These circuit elements are called **comparators** and we have the *comparator circuit value problem*—or **CCV**: "Given a Boolean comparator circuit and binary inputs, is the output = 1?"

The class **CC** is defined as the class of problems log-space reducible to the CCV problem.

Subramanian [192] has shown LFMAXLMATCH to be CC-complete. In addition, he also shows another old friend from earlier in this paper to be CC-complete: the Stable Roommates Problem. For a proof of this, as well as other CC-complete variants, see [196].

There are several interesting open problems surrounding class CC. Is there a *machine* characterization of CC? Are CC and NC comparable?

Feder [87] has shown that CC contains the class NL (non-deterministic log-space). (See once again [22] for a description of this class.)

It seems that there are no results on the parallel complexity or indeed any known

relationships to classes inside P (e.g., CC) for lexicographically first maximum matching.

Let us insert here a short remark on the parallel complexity of MATROID PARITY introduced and discussed in Section 3. Very recently, Narayanan, Saran and V.V. Vazirani [197] have shown that if the matroids are *linear*, there is an RNC^2 algorithm to solve the problem which uses $O(n^{4.5})$ processors.

To close this section, let mention a very recent result on *approximating* a maximum matching in parallel. Fischer, Goldberg and Plotkin [198] have developed an NC^3 algorithm which, given a constant $k > 0$, computes a matching of cardinality at least $1 - 1/(k+1)$ times the maximum. The algorithm requires $O(n^{2k+2})$ processors.

6. Vertex Packing in Parallel

The problems considered here have matching analogs which were discussed in the preceding section. In particular, we will treat the maximal independent set problem (MAXLVP) and its lex-first cousin LFMAXLVP.

Let us dispose of the latter problem first. LFMAXLVP was shown to be P-complete under log-space reduction by Cook in 1985 [64]. (See also [25,26].)

On the other hand, MAXLVP was first shown to be in NC by Karp and Wigderson [159,160] in 1984. They gave an NC^4 algorithm using $O(n^3/(\log n)^3)$ processors. In the same paper, they also provided a faster randomized version, thus showing the problem to be in RNC^3. (This result automatically placed LFMAXLMATCH in the same two classes; this was discussed in Section 5.)

It should be noted that the Karp–Wigderson paper was a deep piece of work. In particular, in order to go from the randomized version of their algorithm to the deterministic version, they appealed to the theory of block designs, no less!

Two independent improvements to NC^2 are due independently to Luby [164,165] and to Alon, Babai and Itai [199]. Both papers first present a randomized version of their respective algorithms and then cleverly remove randomness.

In the past five years or so, several papers have appeared all with their main theme being improvement of the Karp–Wigderson time–processor product bound. Goldberg and Spencer first found an NC^4 algorithm using $O(n)$ processors [200,201] and then produced an NC^3 algorithm requiring $O((n+m)/\log n)$ processors [202]. Preceding both these results, Goldberg had produced a parallel algorithm whose time–processor product was better than Luby or Karp and Wigderson, but the algorithm was not polylog time [203].

The Goldberg–Spencer papers also asked an interesting new question. Turán [204] proved in 1954 that every graph with n vertices and m edges must contain an independent set of size at least $n^2/(2m+n)$. Can such a set be found via an NC algorithm? In [205], they answered their own question in the affirmative, giving an NC^3 algorithm.

Returning now to MAXLVP, let us note that a more efficient NC algorithm for the special case when G is *planar* was found by Xin He [206] who found an NC^2 routine requiring only $O(n)$ processors. In 1988, this result was extended by Khuller [207] to $K_{3,3}$-free graphs. Khuller made use of a special decomposition technique developed by V.V. Vazirani [75,76] in his work on $\Phi(G)$ for $K_{3,3}$-free graphs which we will meet again in the final section of this paper.

Last year, Dadoun and Kirkpatrick [208] reduced the time bound for planar graphs to $O(\log n \log^* n)$, while preserving the $O(n)$ processor bound. (Here $\log^* n$ denotes the number of applications of the log function required to reduce n to a constant value.) They also gave an RNC^1 algorithm for MAXLVP.

Now what about the MAXMVP problem? In other words, what can be said about the (apparently much harder) task of computing a maximum independent set in parallel?

NC algorithms have been found for interval graphs by Helmbold and Mayr [209] and Bertossi and Bonuccelli [210]) and more recently for the more general class of chordal graphs by Naor, Naor and Schäffer [211,212]. To date, these are the only special classes of graphs known to the author for which MAXMVP has been shown to be in NC.

Finally, we also note that an NC algorithm for listing *all* maximal independent sets in a chordal graph is also given in [211,212]. Dahlhaus and Karpinski [213] have now developed NC algorithms for listing *all* maximal independent sets for other classes of graphs, all of which have polynomially bounded numbers of maximal independent sets. (Chordal graphs have this property, for example.)

7. Enumeration

In this section, we will give a short summary of three types of problems dealing with enumeration of matchings and independent sets:

1. Exact counting
2. Approximate counting
3. Complete listing.

Let us begin with counting perfect matchings (#PM) and a sobering result. At this point, the reader will recall the complexity class #P from Section 2. Valiant [66,67] proved the important (but somewhat deflating) result that counting the number of perfect matchings in a bipartite graph (i.e., determining $\Phi(G)$) is #P-complete and therefore NP-hard. It is important to realize that here we have a problem in P (?PM or FINDPM) the counting version of which is #P-hard! It is therefore not surprising that counting maximum independent sets (#MAXMVP) is also #P-complete. So the best we can reasonably hope for in the way of polynomial algorithms for exact counting is to be able to count these objects for certain special classes of graphs.

For matchings, Kasteleyn was the pioneer. Motivated by a question from crystal physics (counting so-called "dimers" on a crystal lattice), he showed [214,215] that there is a polynomial algorithm to count the number of perfect matchings in any *planar* graph. His method involves showing that any undirected planar graph can have its edges oriented in such a way that a certain matrix associated with the resulting directed graph has the property that its determinant is the square of $\Phi(G)$. Such an orientation has come to be called *Pfaffian* after the classical Pfaffian function of a matrix. Since determinants can be evaluated in polynomial time, we have our desired result: a polynomial algorithm for #PM.

Subsequently, Kasteleyn's method was extended to a wider class of graphs by Little [216] who showed that every $K_{3,3}$-free graph has a Pfaffian orientation, although he did not deal with the problem of finding a polynomial algorithm to construct one. In 1988, V.V. Vazirani [75,76] showed how to construct such a polynomial algorithm. In fact, finding a Pfaffian orientation (and hence determining $\Phi(G)$) in a $K_{3,3}$-free graph is even in NC^3! In the same paper, Vazirani showed that testing to see if a graph is $K_{3,3}$-free is also in NC.

Note that a trivial corollary of Vazirani's work shows now that ?PM for $K_{3,3}$-free graphs is in NC. But the nagging problem of FINDPM remains open, even for $K_{3,3}$-free graphs.

Finally, Vazirani showed that the *decision* version of the problem EXACTMATCH, introduced in Section 3, also lies in NC, for the family of $K_{3,3}$-free graphs. Again, we emphasize that the *search* problem remains open.

But let us return for a moment to the problem of finding a Pfaffian orientation for a graph in general. One can pose three (at least formally) different questions here:

1. Does a given graph G have a Pfaffian orientation?

2. Given an orientation of graph G, is it Pfaffian?

3. Given a graph G, find a Pfaffian orientation.

Observe that the first two questions are decision problems, while the third is a search problem. The complexity of all three questions is currently unknown. That question 2 is in co-NP follows from the work of Kasteleyn [214,215]. More recently, Lovász (implicitly) [217] and V.V. Vazirani and Yannakakis [218,219] (explicitly, using Lovász's polynomial time algorithm for computing the GF[2] rank of the set of perfect matchings of a graph) have shown that problems 1 and 2 are polynomially equivalent and hence problem 1 is also in co-NP.

It has been pointed out to us by Pulleyblank [220] that it also follows from the work of Lovász on the matching lattice [217] that problem 3 is equivalent to problems 1 and 2.

In the case when graph G is ***bipartite***, there is an interesting connection between these three problems and a fourth problem which has been studied by Seymour and Thomassen [221], among others.

4. Given a directed graph D, does it fail to contain a directed cycle of even length?

Vazirani and Yannakakis have shown [218,219] that, for bipartite graphs, problem 4 is polynomially equivalent to problems 1 and 2 (and hence problem 3, as well, by Pulleyblank's remark).

Incidently, although #PM is in P for planar graphs by Kasteleyn's work, it has been shown more recently by Jerrum [222] that counting *all* matchings in a planar graph G is #P-complete. Going back to crystal lattices for a moment, Kasteleyn's method of course applies only to those which are planar. What about counting dimers on 3-dimensional lattices? Even this is unknown. (Cf. Sinclair [223].)

Let us also mention at this point that Irving and Leather [224] proved in 1986 that counting the number of stable marriages (i.e., induced bipartite matchings) is #P-complete.

We now proceed from *exact* counting to *approximate* counting. The fact that exact

counting is #P-complete has lent impetus to trying for a less ambitious goal. Can we find a "good approximation" for $\Phi(G)$ "efficiently"?

Here many—but not all—the interesting results have been obtained for the bipartite case. The reason is that evaluating $\Phi(G)$ for a bipartite graph G is the same problem as evaluating the *permanent* of a square (0–1) matrix A. Let A be any $n \times n$ matrix. The **permanent** of A, denoted per A, is just the sum of all the $n!$ terms of its determinant, except all terms are taken with a plus sign. Although the permanent is therefore even slightly easier than the determinant to define, it is a much more badly behaved function! (See Minc [225] for a general reference on permanents.) In particular, although polynomial algorithms for evaluating the determinant are well-known to every undergraduate mathematics student, the best known algorithm for evaluating the permanent has the ugly time bound of $O(n2^n)$! (See Ryser [226].)

Fortunately, there has very recently appeared an excellent survey by Luby [227] dealing with approximation algorithms for the permanent and we heartily recommend that the interested reader consult it. Our remarks on approximation will therefore be brief.

An (ϵ, δ) **approximation algorithm** for per A is a randomized (Monte Carlo) algorithm which accepts an $n \times n$ matrix A and two positive real numbers, ϵ and δ. The algorithm then outputs a number Y as an estimate for per A in the sense that:

$$Prob[(1-\epsilon)\text{per } A \leq Y \leq (1+\epsilon)\text{per } A] \geq 1-\delta.$$

An (ϵ, δ) approximation algorithm is said to be a fully-polynomial **randomized approximation scheme (fpras)** if its running time is polynomial in $n, 1/\epsilon$ and $1/\delta$.

It is an open question as to whether or not there exists a fpras for the permanent function, and therefore for $\Phi(G)$, for G bipartite.

Very recently, two major lines of research on this question have been undertaken. The first of these has resulted in an approximation algorithm which meets the accuracy demand of an fpras, but in superpolynomial time. More specifically, Karmarkar, Karp, Lipton, Lovász and Luby [228] have designed a Monte Carlo algorithm which yields the desired output in time $2^{n^{1/2}}(1/\epsilon^2)\log(1/\delta)p(n)$, where $p(n)$ is a polynomial in n. For fixed ϵ and δ this is about the square root of the time bound for Ryser's algorithm. (Even more recently, Jerrum and U. Vazirani [229] have designed a different algorithm with worst-case time complexity $\exp(O(n^{1/2}\log^2 n))$, which improves that of [228].)

The five authors of [228] also pose the following open question: Is there a *deterministic* algorithm with running time $o(2^n)$ which accepts as input matrix A and a positive real ϵ and outputs Y such that

$$(1-\epsilon)\text{per } A \leq Y \leq (1+\epsilon)\text{per } A \ ?$$

The second approach which seems to have originated with an idea of Broder [230] and been completed by Jerrum and Sinclair [231,232]. The latter two authors succeeded in finding a fpras for *dense* permanents, that is for dense bipartite graphs. The Jerrum-Sinclair papers (and several other companion papers) are not only important for this result, but perhaps even more so for introducing novel approaches using such esoteric concepts from probability theory as rapidly mixing Markov chains and conductance. (Broder too

deals with slightly modified versions of the same.) In general, their common approach deals with reducing the problem of approximately counting perfect matchings to that of generating them at random from an almost uniform distribution.

Using these ideas, Dagum, Luby, Mihail and U.V. Vazirani [233,234] (see also Dagum and Luby [235]) have achieved a polynomial speed up of the algorithm of Jerrum and Sinclair and used it to show that there is also a fpras for bipartite graphs with *large factor size*. The factor size of a bipartite graph $G = A \cup B$ (where $|A| = |B| = n$) is the maximum number of edge-disjoint perfect matchings in G. (Observe that a graph with an αn-factor must have minimum degree at least αn, but not necessarily vice-versa.) In this area, Dagum and Luby have shown that there is a fpras for bipartite graphs with factor size at least αn for any constant $\alpha > 0$.

These results are even more interesting when compared to some related new completeness results. Broder [230] has shown that *exact* counting in dense graphs is as hard as exacting counting in general and so is #P-complete. Dagum and Luby [235] and Dagum, Luby, Mihail and U.V. Vazirani [233,234] have shown that exact counting in $f(n)$-regular bipartite graphs is #P-complete for any $f(n)$ such that $3 \leq f(n) \leq n - 3$. In fact, they show that for any $\epsilon > 0$ and for any function $f(n)$ such that $3 \leq f(n) \leq n^{1-\epsilon}$, the existence of a fpras for $f(n)$-regular bipartite graphs would imply the existence of a fpras for *all* bipartite graphs!

And finally a word or two about *finding* (i.e., *listing*) *all* matchings and independent sets. Indeed the literature is quite sparse for these problems.

For bipartite graphs, Grigoriev and Karpinski [191] have developed algorithms for constructing all maximum matchings in the case when $\Phi(G)$ is bounded. More particularly, they show #PM is in NC^2 when $\Phi(G)$ is bounded by a constant and in NC^3 when $\Phi(G)$ is polynomially bounded (i.e., $\Phi(G) \leq cn^k$). They claim their results extend to *non-bipartite* graphs as well, although they give no proofs.

For bipartite graphs in general, it appears that the best algorithm for finding all perfect matchings is due to Fukuda and Matsui [236] who found an $O((\Phi(G)+1)(n+m+n^{2.5}))$ (sequential) algorithm for the problem. We are unaware of any published parallel algorithms in this case.

For independent set listing problems, even less seems to be known. The sole result we will mention is the following. Let M denote the number of maximal independent sets in a graph. The fastest *sequential* algorithm for constructing *all* maximal independent sets in the graph has $O(nmM)$ time and $O(n+m)$ space. It is due independently to Chiba and Nishizeki [237] and Tsukiyama, Ide, Ariyoshi and Shirakawa [238]. Of course, this is a *polynomial* algorithm if M is polynomially bounded.

More recently, Dahlhaus and Karpinski [213] have obtained a parallel algorithm for this problem which runs in $O(\log^3(nM))$ time and uses $O(M^6 n^2)$ processors. Note that if M is polynomially bounded, this becomes an NC algorithm. There are many interesting classes of graphs with bounded M and the authors give a list of eight such families.

8. Lower Bounds

Note that all of the considerable work discussed and/or referred to above in this

paper possesses one common thread: Can we find a *faster* algorithm to solve the problem at hand?

Much less progress has been made on approaches made from the opposite direction: Can we prove that *no* polynomial time algorithm is possible for a given problem? Or, more generally, can we prove that no algorithm is possible in a given time or space for a given problem?

Though results of this kind—that is, finding *lower* bounds for the computational complexity of certain problems—are much more sparse, we will close this paper with a brief overview of the work of A.A. Razborov in this area. His work has caused considerable excitement among complexity theorists world-wide and has won him the Nevanlinna Prize at the International Congress of Mathematicians held in Kyoto in 1990. For our brief synopsis, we borrow heavily from the articles of Lovász [239] and Sipser [45].

This work is cast in the terminology of boolean circuit theory. (We mentioned boolean circuits briefly in our treatment of the complexity class CC in Section 5.) A **boolean circuit** for a computation is an acyclic directed graph the nodes of which represent the elementary steps in the computation. These nodes are also called *gates*. Gates having indegree 0 are called *input* gates and those with outdegree 0 are *output* gates. The simplest kinds of gates are AND, OR and NOT gates. The **size** of a boolean circuit is the number of gates in it. Every polynomial time computable function can be computed by a polynomial size boolean circuit. More precisely this means that if L is a language in P and L_n denotes the set of strings of length n in L, then there exists a family B_n of boolean circuits such that B_n takes n input bits and recognizes L_n (i.e., outputs 1 if and only if the n-bit input string is in L_n); and the size of B_n is bounded by n^c for some fixed constant c.

So if we could prove that a superpolynomial lower bound exists on the circuit size required by some NP problem such as the existence of a clique of given size, then this would suffice to separate NP from P.

Now let us define a boolean circuit to be **monotone** if it contains no NOT gates. Razborov, in two 1985 papers, showed that neither (a) [240] deciding the existence of a clique of given size in a graph nor (b) [241] deciding the existence of a perfect matching in a bipartite graph (a special case of (a)) can be done using polynomial-size *monotone* circuits. In particular, he showed that at least $n^{(c \log n)}$ gates are necessary for a monotone circuit solution of perfect matching. So ?PM, which is well-known to have polynomial non-monotone complexity, has *super-polynomial* monotone complexity. Alon and Boppana [242] have since strengthened Razborov's clique result to show that in fact there is an *exponential* lower bound for the clique problem and not just a superpolynomial bound.

The clique problem is clearly equivalent to the independent set problem via graph complementation. Hence Razborov's results deal precisely with matching and vertex packing, our two paradigm problems. Will monotone circuits somehow prove to yield the key to the apparent separation in the complexity of these two problems and perhaps even provide the answer to the P = NP question? Only time will tell.

This brings us to the end of our survey. As is always the case, more could have been included. There are several important topics related to the theme of this paper, but which we have elected not to include. Among these are *weighted* and capacitated matching. For useful references, see [243,244,245,246,247,248]. For *probabilistic* analysis

of graph algorithms, see [249,250,251,252,253]. And for matching and vertex packing in *random* graphs, see [254,255,256,257,258,259,260,261,262].

Acknowledgements

It is a pleasure to acknowledge the kind assistance of a number of colleagues. Copious thanks are due to R. Anstee, L. Babai, G. Brassard, E. Eschen, A. Frieze, Z. Galil, A.B. Gamble, P. Hell, M. Jerrum, D.S. Johnson, R.M. Karp, L. Lovász, M. Luby, J. Naor, W.R. Pulleyblank, R. Motwani, E.C. Sewell, M. Sipser, L.J. Stockmeyer, É. Tardos, C. Thomassen, L. Trotter and V.V. Vazirani. We are very grateful for their help.

Notes added in proof: Since the completion of this paper, reference [263] has come to the attention of the author and should be mentioned in Section 5 along with references [150,151,152,153,154]. Similarly, reference [264] should be added to references [237,238] in Section 7. In [264], the authors present an algorithm which outputs all maximal independent sets *in lexicographic order.*

References

[1] L. Lovász and M.D. Plummer, *Matching Theory*, **29**, Ann. Discrete Math., North-Holland, Amsterdam, 1986.

[2] M.D. Plummer, Matching theory – a sampler: from Dénes Kõnig to the present, preprint, 1991.

[3] D. Kõnig, Graphs and matrices, *Mat. Fiz. Lapok*, **38**, 1931, 116-119. (Hungarian)

[4] _____, Über trennende Knotenpunkte in Graphen (nebst Anwendungen auf Determinanten und Matrizen, *Acta Sci. Math. (Szeged)*, **6**, 1933, 155-179.

[5] T. Gallai, Über extreme Punkt- und Kantenmengen, *Ann. Univ. Sci. Budapest, Eötvös Sect. Math.*, **2**, 1959, 133-138.

[6] J. Edmonds, Paths, trees and flowers, *Canad. J. Math.*, **17**, 1965, 449-467.

[7] N. Blum, A new approach to maximum matching in general graphs, *Automata, Languages and Programming*, Ed.: M.S. Paterson, Lecture Notes in Computer Science vol. 443, Springer-Verlag, Berlin, 1990, 586-597.

[8] _____, A new approach to maximum matching in general graphs, Univ. Bonn Inst. für Informatik Report No. 8546-CS, February, 1990.

[9] _____, Jack Edmonds original maximum matching algorithm needs only $O(n^3)$ time, Univ. Bonn Inst. für Informatik Report No. 8568-CS, June, 1991.

[10] S. Micali and V.V. Vazirani, An $O(V^{1/2}E)$ algorithm for finding maximum matching in general graphs, *Proc. 21st Annual Symposium on Foundations of Computer Science*, IEEE, New York, 1980, 17-27.

[11] V.V. Vazirani, A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{V}E)$ general graph matching algorithm, Cornell Univ. Dept. of Computer Sci. Technical Report TR 89-1035, September, 1989.

[12] J.E. Hopcroft and R.M. Karp, An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, *Proc. 12th Annual Symposium on Switching and Automata Theory (East Lansing, 1971)*, IEEE, New York, 1971, 122-125.

[13] _____, An $n^{5/2}$ algorithm for maximum matchings

in bipartite graphs, SIAM J. Comput., **2**, 1973, 225-231.

[14] T. Feder and R. Motwani, Clique partitions, graph compression and speeding-up algorithms, *Proc. 23rd Annual ACM Symposium on Theory of Computing*, ACM, New York, 1991, 123-133.

[15] H. Alt, N. Blum, K. Mehlhorn and M. Paul, Computing a maximum cardinality matching in a bipartite graph in time $O(n^{1.5}\sqrt{m/\log n})$, *Inform Proc. Lett.*, **37**, 1991, 237-240.

[16] J. Cheriyan, T. Hagerup and K. Mehlhorn, Fast and simple network algorithms (extended abstract), preprint, 1991.

[17] R.E. Tarjan and A.E. Trojanowski, Finding a maximum independent set, *SIAM J. Comput.*, 6, 1977, 537-546.

[18] F. Harary, *Graph Theory*, Addison-Wesley, Reading, 1969.

[19] G.J. Minty, On maximal independent sets of vertices in claw-free graphs, *J. Combin. Theory Ser. B*, **28**, 1980, 284-304.

[20] N. Sbihi, Algorithme de recherche d'un stable de cardinalité maximum dans un graphe sans étoile, *Discrete Math.*, **29**, 1980, 53-76.

[21] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., San Francisco, 1979.

[22] D.S. Johnson, A catalog of complexity classes, Chapter 2 in: *Handbook of Theoretical Computer Science Volume A: Algorithms and Complexity*, Ed.: J. van Leeuwen, Elsevier/MIT, Amsterdam/Cambridge, 1990, 69-161.

[23] P. van Emde Boas, Machine models and simulations, Chapter 1 in: *Handbook of Theoretical Computer Science Volume A: Algorithms and Complexity*, Ed.: J. van Leeuwen, Elsevier/MIT, Amsterdam/Cambridge, 1990, 1-66.

[24] R.B. Boppana and M. Sipser, The complexity of finite functions, Chapter 14 in: *Handbook of Theoretical Computer Science Volume A: Algorithms and Complexity*, Ed.: J. van Leeuwen, Elsevier/MIT, Amsterdam/Cambridge, 1990, 759-804.

[25] R.M. Karp and V. Ramachandran, A survey of parallel algorithms for shared-memory machines, Univ. of California at Berkeley, Computer Sci. Div. (EECS) Report No. UCB/CSD 88/408, March, 1988. (has appeared in: Handbook of Theoretical Computer Science, North-Holland, 1990. See ref-

erence [26].)

[26] _____, Parallel algorithms for shared-memory machines, Chapter 17 in: *Handbook of Theoretical Computer Science, Vol. A: Algorithms and Complexity*, Ed.: J. van Leeuwen, Elsevier/MIT Press, Amsterdam/Cambridge, 1990, 869-942.

[27] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass., 1974.

[28] D. Angluin, On counting problems and the polynomial-time hierarchy, *Theoret. Comput. Sci.*, **12**, 1980, 161-173.

[29] R. Anderson, The complexity of parallel algorithms, Stanford Univ. Dept. of Computer Science, Ph.D. Thesis, 1984.

[30] J. L. Balcázar, J. Díaz and J. Gabarró, *Structural Complexity I*, Springer-Verlag, Berlin, 1988.

[31] S.A. Cook, An observation on time-storage trade-off, *J. Comput. System Sci.*, **9**, 1974, 308-316.

[32] _____, An overview of computational complexity, *Comm. ACM*, **26**, 1983, 401-408.

[33] S. Fortune and J. Wyllie, Parallelism in random access machines, *Proc. 10th Annual ACM Symposium on Theory of Computing, (San Diego, May, 1978)*, ACM, New York, 1978, 114-118.

[34] J. Gill, Computational complexity of probabilistic Turing machines, *SIAM J. Comput.*, **6**, 1977, 675-695.

[35] R.M. Karp, The probabilistic analysis of some combinatorial search problems, in: *Algorithms and Complexity*, Ed.: J.F. Traub, Academic Press, New York, 1976, 1-19.

[36] _____, An introduction to randomized algorithms, Internat. Comput. Sci. Inst. Tech. Report TR-90-024, June, 1990. (to appear: *Discrete Appl. Math.*.)

[37] E.M. Eschen, Synchronous parallel computation complexity: an overview, preprint, 1985.

[38] R.M. Karp and M. Luby, Monte-Carlo algorithms for enumeration and reli-

ability problems, *Proc. 24th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1983, 56-64.

[39] C. Lautemann, BPP and the polynomial hierarchy, *Inform. Process. Letters*, **17**, 1983, 215-217.

[40] C. Lund, L. Fortnow, H. Karloff and N. Nisan, Algebraic methods for interactive proof systems, *Proc. 20th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1990, 2-10.

[41] N. Nisan and A. Wigderson, Hardness vs. randomness (Extended Abstract), *Proc. 29th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1988, 2-11.

[42] A. Panconesi and D. Ranjan, Quantifiers and approximation (Extended Abstract, *Proc. 22nd Annual ACM Symposium on Theory of Computing*, ACM, New York, 1990, 446-456.

[43] W.J. Savitch, Relationships between non-deterministic and deterministic tape complexities, *J. Comput. and System Sci.*, **4**, 1970, 177-192.

[44] M. Sipser, A complexity theoretic approach to randomness, *Proc. 15th Annual ACM Symposium on Theory of Computing (Boston, 1983)*, ACM, New York, 1983, 330-335.

[45] _____, Alexander Razborov, *Notices of the Amer. Math. Soc.*, 37, 1990, 1215-1216.

[46] D.B. Shmoys and É. Tardos, Computational complexity, *The Handbook of Combinatorics*, Eds.: R.L. Graham, M. Grötschel and L. Lovász, North-Holland, Amsterdam, (to appear).

[47] L.J. Stockmeyer, The polynomial-time hierarchy, *Theoret. Comput. Sci.*, **3**, 1977, 1-22.

[48] _____, The complexity of approximate counting, *Proc. 15th Annual ACM Symposium on Theory of Computing (Boston, 1983)*, ACM, New York, 1983, 118-126.

[49] _____, On approximation algorithms for $\#P$, *SIAM J. Comput.*, **14**, 1985, 849-861.

[50] _____, Classifying the computational complexity of problems, *J. Symbolic Logic*, **52**, 1987, 1-43.

[51] U.V. Vazirani and V.V. Vazirani, Random polynomial time is equal to slightly-random polynomial time, *26th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1985, 417-428.

[52] D.J.A. Welsh, Problems in computational complexity, in: *Applications of Combinatorics*, Ed.: R.J. Wilson, Shiva Mathematics Series, Shiva Publ. Ltd., Nantwich, 1982, 75-85.

[53] _____, Randomised algorithms, *Discrete Appl. Math.*, **5**, 1983, 133-145.

[54] C. Wrathall, Complete sets and the polynomial-time hierarchy, *Theoret. Comput. Sci.*, **3**, 1977, 23-33.

[55] S. Zachos, Robustness of probabilistic computational complexity classes under definitional perturbations, *Inform. and Control*, **54**, 1982, 143-152.

[56] _____, Probabilistic quantifiers, adversaries and complexity classes: an overview, *Structure in Complexity Theory (Berkeley, 1986)*, Ed.: A.L. Selman, Lecture Notes in Computer Science Vol. **223**, Springer-Verlag, Berlin, 1986, 383-400.

[57] M.D. Plummer, Some covering concepts in graphs, *J. Combin. Theory* , 8, 1970, 91-98.

[58] S.A. Cook, The complexity of theorem-proving procedures, *Proc. 3rd Annual ACM Sympos. on Foundations of Computer Science (Shaker Heights)*, 1971, 151-158.

[59] L.A. Levin, Universal search problems, *Problemy Peredaci Informacii*, 9, 1973, 115-116 (in Russian). (translation: *Problems of Information Transmission*, **9**, 1973, 265-266.

[60] B.A. Trakhtenbrot, A survey of the Russian approach to perebor (brute-force search) algorithms, *Ann. Hist. Comput.*, **6**, 1984, 384-400.

[61] R.M. Karp, Reducibility among combinatorial problems, in *Complexity of Computer Computations*, Eds.: R.E. Miller and J.W. Thatcher, Plenum Press, New York, 1972, 85-103.

[62] D. Dobkin, R.J. Lipton and S. Reiss, Linear programming is log-space hard for P, *Inform. Process. Lett.*, **8**, 1979, 96-97.

[63] S.A. Cook, The classification of problems which have fast parallel algorithms,

Foundations of Computation Theory, (Proc. 1983 Internat. FCT-Conference, Borgholm, Sweden, August, 1983), Ed.: M. Karpinski, Lecture Notes in Computer Science Vol. **158**, Springer-Verlag, Berlin, 1983, 78-93.

[64] _____, A taxonomy of problems with fast parallel algorithms, *Inform. and Control*, **64**, 1985, 2-22.

[65] N. Pippenger, On simultaneous resource bounds, *Proc. 20th Annual Symposium on Foundations of Computer Science*, IEEE, New York, 1979, 307-311.

[66] L.G. Valiant, The complexity of computing the permanent, *Theoret. Comput. Sci.*, **8**, 1979, 189-201.

[67] _____, The complexity of enumeration and reliability problems, *SIAM J. Comput.*, **8**, 1979, 410-421.

[68] M. Yannakakis and F. Gavril, Edge dominating sets in graphs, *SIAM J. Appl. Math.*, **38**, 1980, 364-372.

[69] W.R. Pulleyblank, Matchings and extensions, *Handbook of Combinatorics*, Eds.: R.L. Graham, M. Grötschel and L. Lovász, North-Holland, Amsterdam, (to appear).

[70] A. Itai, M. Rodeh and S.L. Tanimoto, Some matching problems for bipartite graphs, *J. Assoc. Comput. Mach.*, **25**, 1978, 517-525.

[71] J. Edmonds, Maximum matching and a polyhedron with (0,1)-vertices, *J. Res. Nat. Bur. Standards Sect. B*, **69B**, 1965, 125-130.

[72] C. Papadimitriou and M. Yannakakis, The complexity of restricted spanning tree problems, *J. Assoc. Comput. Mach.*, **29**, 1982, 285-309.

[73] C.H. Papadimitriou, Polytopes and complexity, in: *Progress in Combinatorial Optimization*, Ed.: W.R. Pulleyblank, Academic Press, Toronto, 1984, 295-304.

[74] F. Barahona and W.R. Pulleyblank, Exact arborescences, matchings and cycles, *Discrete Appl. Math.*, **16**, 1987, 91-99.

[75] V.V. Vazirani, NC algorithms for computing the number of perfect matchings in $K_{3,3}$-free graphs and related problems, *SWAT 88; Proc. First Scandinavian Workshop on Algorithm Theory (Halmstad, July, 1988)*, Eds.: R. Karlson and A. Lingas, Lecture Notes in Computer Science Vol. **318**, Springer-Verlag, Berlin, 1988, 233-242.

[76] _____, NC algorithms for computing the number of perfect matchings in $K_{3,3}$-free graphs and related problems, *Inform. and Comput.*, 80, 1989, 152-164.

[77] W.R. Pulleyblank, Alternating cycle free matchings, preprint, 1982.

[78] L.J. Stockmeyer and V.V. Vazirani, NP-completeness of some generalizations of the maximum matching problem, **15**, 1982, 14-19.

[79] K. Cameron, Induced matchings, *Discrete Appl. Math.*, **24**, 1989, 97-102.

[80] S. Even, O. Goldreich and P. Tong, On the NP-completeness of certain network-testing problems, Technion, Haifa, Dept. Comput. Sci. Tech. Rpt. #230, 1981.

[81] D. Gusfield and R.W. Irving, *The Stable Marriage Problem: Structure and Algorithms*, MIT Press, Cambridge, 1989.

[82] D. Gale and L.S. Shapley, College admissions and the stability of marriage, *Amer. Math. Monthly*, **69**, 1962, 9-15.

[83] R.W. Irving, An efficient algorithm for the stable roommates problem, *J. Algorithms*, **6**, 1985, 577-595.

[84] C. Ng and D.S. Hirschberg, Lower bounds for the stable marriage problem and its variants, *SIAM J. Comput.*, **19**, 1990, 71-77.

[85] E. Ronn, On the complexity of stable matchings with and without ties, Yale Univ. Dept. of Computer Sci. Ph.D. Thesis, 1986.

[86] _____, NP-complete stable matching problems, *J. Algorithms*, **11**, 1990, 285-304.

[87] T. Feder, A new fixed point approach for stable networks and stable marriages, *Proc. 21st Annual ACM Symposium on Theory of Computing*, ACM, New York, 1989, 513-522.

[88] D.G. Kirkpatrick and P. Hell, On the completeness of a generalized matching problem, *Proc. 10th Annual ACM Symposium on Theory of Computing (San Diego, May, 1978*, ACM, New York, 1978, 240-245.

[89] _____, On the complexity of general graph factor problems, *SIAM J. Comput.*, **12**, 1983, 601-609.

[90] P. Hell and D.G. Kirkpatrick, Scheduling, matching, and coloring, *Algebraic Methods in Graph Theory, Szeged (Hungary), 1978*, Colloq. Math. Soc. János Bolyai, **25**, 1978, 273-279.

[91] _____, Packing by cliques and by finite families of graphs, *Discrete Math.*, **49**, 1984, 45-59.

[92] G. Cornuéjols, D. Hartvigsen and W.R. Pulleyblank, Packing subgraphs in a graph, *O.R. Letters*, **1**, 1982, 139-143.

[93] G. Cornuéjols and W.R. Pulleyblank, Perfect triangle-free 2-ma[illegible]igs, *Combinatorial Optimization II (Proc. Conf. Univ. East Anglia, Norwich, 1979)*, Math. Programming Stud. No. 13, North-Holland, Amsterdam, 1980, 1-7.

[94] _____, A matching problem with side conditions, *Discrete Math.*, **29**, 1980, 135-159.

[95] P. Hell and D.G. Kirkpatrick, On generalized matching problems, *Inform. Process. Lett.*, **12**, 1981, 33-35.

[96] _____, Packings by complete bipartite graphs, *SIAM J. Alg. Disc. Meth.*, **7**, 1986, 199-209.

[97] P. Hell, D. Kirkpatrick, J. Kratochvíl and I. Kříž, On restricted two-factors, SIAM J. Disc. Math., **1**, 1988, 472-484.

[98] F. Berman, D. Johnson, T. Leighton, P.W. Shor and L. Snyder, Generalized planar matching, *J. Algorithms*, **11**, 1990, 153-184.

[99] M.E. Dyer and A.M. Frieze, Planar 3DM is NP-complete, *J. Algorithms*, **7**, 1986, 174-184.

[100] B.S. Baker, Approximation algorithms for NP-complete problems on planar graphs, *Proc. 24th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1983, 265-273.

[101] M.W. Padberg and M.R. Rao, Odd minimum cut-sets and b-matchings, *Math. Oper. Res.*, **7**, 1982, 67-80.

[102] F. Barahona, Reducing matching to polynomial size linear programming, Univ. Waterloo Dept. of Combinatorics and Optimization Res. Report CORR88-51, 1988.

[103] _____, On cuts and matchings in planar graphs, Univ. Bonn Inst. für

Ökonometrie und Oper. Res. Report 88503-OR, January, 1988.

[104] A.B. Gamble, Polyhedral extensions of matching theory, Univ. of Waterloo Dept. of Combinatorics and Optimization Ph.D. Thesis, 1989.

[105] M. Yannakakis, Expressing combinatorial optimization problems by linear programs, *Proc. 20th Annual ACM Symposium on Theory of Computing*, ACM, New York, 1988, 223-228.

[106] E.L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

[107] L. Lovász, The matroid matching problem, *Algebraic Methods in Graph Theory II*, Eds.: L. Lovász and V.T. Sós, Colloq. Soc. János Bolyai, **25**, 1981, 495-517.

[108] H.N. Gabow and M. Stallmann, An augmenting path algorithm for the parity problem on linear matroids, *Proc. 25th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1984, 217-228.

[109] S. Poljak, A note on stable sets and coloring of graphs, *Comment. Math. Univ. Carolin.*, **15**, 1974, 307-309.

[110] M.R. Garey, D.S. Johnson and L. Stockmeyer, Some simplified NP-complete graph problems, *Theoret. Comput. Sci.*, **1**, 1976, 237-267.

[111] F.B. Shepherd, Near-perfection and stable set polyhedra, Univ. of Waterloo, Dept. of Combinatorics and Optimization Ph.D. Thesis, 1990.

[112] N.V.R. Mahadev, Stability numbers in structured graphs, Univ. of Waterloo, Dept. of Combinatorics and Optimization Ph.D. Thesis, 1984.

[113] C. Berge, Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind, *Wiss. Zeitung, Martin Luther Univ. Hall-Wittenberg*, 1961, 114.

[114] L. Lovász, Normal hypergraphs and the weak perfect graph conjecture, *Discrete Math.*, **2**, 1972, 253-267.

[115] M. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[116] C. Berge and V. Chvátal (Eds.), *Topics on Perfect Graphs*, Ann. Discrete Math. Vol. **21**, 1984, North-Holland, Amsterdam.

[117] M. Grötschel, L. Lovász and A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization, *Combinatorica*, **1**, 1981, 169-197.

[118] _____, Polynomial algorithms for perfect graphs, *Topics on Perfect Graphs*, Eds.: C. Berge and V. Chvátal, Ann. Discr. Math., **21**, 1984, 325-356.

[119] _____, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, Berlin, 1988.

[120] _____, Relaxations of vertex packing, *J. Combin. Theory Ser. B*, **40**, 1986, 330-343.

[121] L. Lovász, Some finite basis theorems in graph theory, *Combinatorics*, II, Eds.: A. Hajnal and V.T. Sós, Colloq. Math. Soc. János Bolyai, **18**, North-Holland, Amsterdam, 1978, 717-729.

[122] _____, Vertex packing algorithms, *Automata, Languages and Programming (Nafplion, Greece, 1985)*, Ed.: W. Brauer, Lecture Notes in Computer Science Vol. **194**, Springer-Verlag, Berlin, 1985, 1-14.

[123] _____, Stable sets and polynomials, preprint, Princeton Univ. Dept. of Comput. Sci., June, 1990.

[124] L. Lovász and A. Schrijver, Matrix cones, projection representations, and stable set polyhedra, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 1, Amer. Math. Soc., Providence, 1990, 1-17.

[125] F. Sterboul, A characterization of the graphs in which the transversal number equals the matching number, *J. Combin. Theory Ser. B*, **27**, 1979, 228-229.

[126] R.W. Deming, Independence numbers of graphs — an extension of the König-Egerváry theorem, *Discrete Math.*, **27**, 1979, 23-33.

[127] L. Lovász, Ear-decompositions of matching-covered graphs, *Combinatorica*, **3**, 1983, 105-117.

[128] E. Korach, On dual integrality, min-max equalities and algorithms in combinatorial programming, Univ. of Waterloo Dept. of Combinatorics and Optimization Ph.D. Thesis, 1982.

[129] G.L. Nemhauser and L.E. Trotter, Vertex packings: structural properties and algorithms, *Math. Programming*, **8**, 1975, 232-248.

[130] W.R. Pulleyblank, Minimum node covers and 2-bicritical graphs, *Math. Programming*. **17**, 1979, 91-103.

[131] J.-M. Bourjolly and W.R. Pulleyblank, Kõnig-Egerváry graphs, 2-bicritical graphs and fractional matchings, *Discr. Appl. Math.*, **24**, 1989, 63-82.

[132] V. Chvátal and P.J. Slater, A note on well-covered graphs, preprint, January, 1991. (to appear in the Proceedings of the Quo Vadis Conf., Univ. Alaska at Fairbanks, July, 1990.)

[133] R.S. Sankaranarayana and L. K. Stewart, Complexity results for well covered graphs, Univ. of Alberta Dept. of Computing Science Tech. Report TR 90-21, August, 1990.

[134] M.D. Plummer, On well-covered graphs — a survey, preprint, 1991.

[135] N. Dean and J. Zito, Well-covered graphs and extendability, preprint, December 13, 1990.

[136] A. Hajnal, A theorem on k-saturated graphs, *Canad. J. Math.*, **17**, 1965, 720-724.

[137] B. Andrásfai, On critical graphs, *Theory of Graphs (International Symposium, Rome, 1966)*, Ed.: P. Rosenstiehl, Gordon and Breach, New York, 1967, 9-19.

[138] V. Chvátal, On certain polytopes associated with graphs, *J. Combin. Theory Ser. B*, **18**, 1975, 138-154.

[139] E.C. Sewell, Stability critical graphs and the stable set polytope, Cornell Computational Optimization Project, Cornell Univ., Tech. Report 90-11, May, 1990.

[140] _____, Stability critical graphs and the stable set polytope, Cornell Univ. School of Oper. Res. and Indust. Eng. Tech. Report 905, May, 1990.

[141] E.C. Sewell and L.E. Trotter, Jr., Stability critical graphs and even subdivisions of K_4, Cornell Univ. School of ORIE Preprint, June, 1990.

[142] C.H. Papadimitriou and M. Yannakakis, The complexity of facets (and some facets of complexity), *Proc. 14th Annual ACM Symposium on Theory of Computing*, ACM, New York, 1982, 255-260.

[143] C.H. Papadimitriou and D. Wolfe, The complexity of facets resolved, *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1985, 74-78.

[144] U. Feige, S. Goldwasser, L. Lovász and S. Safra, On the complexity of approximating the maximum size of a clique, preprint, November, 1990.

[145] U. Feige, S. Goldwasser, L. Lovász, S. Safra and M. Szegedy, Approximating clique is almost NP-complete (extended abstract), preprint, April, 1991.

[146] _____, Approximating clique is almost NP-complete, *Proc. 21st Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1991, (to appear).

[147] R. Boppana and M.M. Halldórsson, Approximating maximum independent sets by excluding subgraphs, *SWAT 90 (Bergen, Sweden)*, Eds.: J.R. Gilbert and R. Karlsson, Lecture Notes in Computer Science Vol. **447**, 1990, 13-25.

[148] R. Bar-Yehuda and S. Even, A $2 - (\log\log n / \log n)$ performance ratio for the weighted vertex cover problem, Technion, Haifa, Tech. Report 260, January, 1983.

[149] B. Monien and E. Speckenmeyer, Ramsey numbers and an approximation algorithm for the vertex cover problem, *Acta Inform.*, **22**, 1985, 115-123.

[150] H.N. Gabow and R.E. Tarjan, Almost-optimum speed-ups of algorithms for bipartite matching and related problems, *Proc. 20th Annual ACM Symposium on Theory of Computing*, ACM, New York, 1988, 514-527.

[151] A.V. Goldberg, S.A. Plotkin and P.M. Vaidya, Sublinear-time parallel algorithms for matching and related problems, *Proc. 29th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1988, 174-185.

[152] H.N. Gabow and R.E. Tarjan, Almost-optimum parallel speed-ups of algorithms for bipartite matching and related problems, Princeton Univ. Dept. of Computer Sci. Report CS-TR-223-89, January, 1989.

[153] Y. Shiloach and U. Vishkin, An $O(n^2 \log n)$ parallel MAX-FLOW algorithm, *J. Algorithms*, **3**, 1982, 128-146.

[154] T. Kim and K-Y. Chwa, An $O(n \log n \log\log n)$ parallel maximum matching algorithm for bipartite graphs, *Inform. Proc. Lett.*, **24**, 1987, 15-17.

[155] G. Lev, Size bounds and parallel algorithms for networks, Univ. of Edinburgh Dept. of Computer Sci. Report CST-8-80, (Ph.D. Thesis), September, 1980.

[156] G. Lev, N. Pippenger and L.G. Valiant, A fast parallel algorithm for routing in permutation networks, *IEEE Trans. on Computers*, C-30, 1981, 93-100.

[157] E. Dahlhaus and M. Karpinski, Perfect matching for regular graphs is AC^0-hard for the general matching problem, preprint, 1990.

[158] A. Israeli and Y. Shiloach, An improved parallel algorithm for maximal matching, *Inform. Process. Lett.*, **22**, 1986, 57-60.

[159] R.M. Karp and A. Wigderson, A fast parallel algorithm for the maximal independent set problem, *Proc. 16th Annual ACM Symposium on Theory of Computing, ACM, New York, 1984, 266-272.*

[160] _____, A fast parallel algorithm for the maximal independent set problem, *J. Assoc. Comput. Mach.*, 32, 1985, 762-773.

[161] A. Israeli and A. Itai, A fast and simple randomized parallel algorithm for maximal matching, *Inform. Process. Lett.*, **22**, 1986, 77-80.

[162] M. Luby, Removing randomness in parallel computation without a processor penalty (Preliminary Version), *Proc. 29th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Soc. Press, 1988, 162-173.

[163] _____, Removing randomness in parallel computation without a processor penalty, Internat. Computer Sci. Institute Tech. Report TR-89-044, July, 1989.

[164] _____, A simple parallel algorithm for the maximal independent set problem, *Proc. 17th Annual ACM Symposium on Theory of Computing*, ACM, New York, 1985, 1-10.

[165] _____, A simple parallel algorithm for the maximal independent set problem, *SIAM J. Comput.*, **15**, 1986, 1036-1053.

[166] R.M. Karp, E. Upfal and A. Wigderson, Are search and decision problems computationally equivalent?, *Proc. 17th Annual ACM Symposium on Theory of Computing*, ACM, New York, 1985, 464-475.

[167] L. Lovász, On determinants, matchings and random algorithms, *Fundamen-*

tals of Computation Theory, FCT '79, (Proc. Conf. Algebraic, Arithmetic and Categorical Methods in Computation Theory, Berlin/Wendisch-Rietz 1979), Ed.: L. Budach, Math. Research 2, Akademie-Verlag, Berlin, 1979, 565-574.

[168] A. Borodin, J. von zur Gathen and J. Hopcroft, Fast parallel matrix and GCD computations, *Proc. 23rd Symposium on Theory of Computing*, ACM, New York, 1982, 65-71.

[169] _____, Fast parallel matrix and GCD computations, *Inform. and Control*, **52**, 1982, 241-256.

[170] L. Csánky, Fast parallel matrix inversion algorithms, *SIAM J. Comput.*, **5**, 1976, 618-623.

[171] R.M. Karp, E. Upfal and A. Wigderson, Constructing a perfect matching is in random NC, *Proc. 17th Annual ACM Symposium on Theory of Computing (Providence, Rhode Island)*, ACM, New York, 1985, 22-32.

[172] _____, Constructing a perfect matching is in random NC, *Combinatorica*, **6**, 1986, 35-48.

[173] Z. Galil and V. Pan, Improved processor bounds for combinatorial problems in RNC, *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1985, 490-495.

[174] _____, Improved processor bounds for combinatorial problems in RNC, *Combinatorica*, **8**, 1988, 189-200.

[175] K. Mulmuley, U.V. Vazirani and V.V. Vazirani, Matching is as easy as matrix inversion, *Proc. 19th Annual ACM Symposium on Theory of Computing*, ACM, New York, 1987, 345-354.

[176] _____, Matching is as easy as matrix inversion, *Combinatorica*, **7**, 1987, 105-113.

[177] H.J. Karloff, A Las Vegas RNC algorithm for maximum matching, *Combinatorica*, **6**, 1986, 387-391.

[178] Z. Galil, Sequential and parallel algorithms for finding maximum matchings in graphs, *Ann. Rev. Comput. Sci.*, **1**, 1986, 197-224.

[179] _____, Efficient algorithms for finding maximum matching in graphs, *Computing Surveys*, **18**, 1986, 23-38.

[180] M. Chrobak and J. Naor, Computing a perfect matching in claw-free graphs, preprint, 1990.

[181] J. Naor, Computing a perfect matching in a line graph, *VLSI Algorithms and Architectures, 3rd Aegean Workshop on Computing, AWOC 88*, Ed.: J.H. Reif, Lecture Notes in Computer Sci. Vol. **319**, Springer-Verlag, Berlin, 1989, 139-148.

[182] G.L. Miller and J. Naor, Flow in planar graphs with multiple sources and sinks, *Proc. 30th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1989, 112-117.

[183] E. Dahlhaus and M. Karpinski, Parallel construction of perfect matchings and Hamiltonian cycles on dense graphs, *Theoret. Comput. Sci.*, **61**, 1988, 121-136.

[184] E. Dahlhaus, P. Hajnal and M. Karpinski, Optimal parallel algorithm for the Hamiltonian cycle problem on dense graphs, *Proc. 29th Annual IEEE Symposium on Foundations of Computer Science (White Plains)*, 1988, 186-193.

[185] _____, On the parallel complexity of Hamiltonian cycle and matching problems on dense graphs, preprint, 1990.

[186] E. Dahlhaus and M. Karpinski, The matching problem for strongly chordal graphs is in NC, Univ. Bonn Inst. für Informatik Research Report No. 855-CS, December, 1986.

[187] D. Helmbold and E. Mayr, Applications of parallel scheduling to perfect graphs, *Proc. Internat. Workshop WG '86*, Lecture Notes in Computer Science Vol. **246**, 1987, 188-203.

[188] _____, Applications of parallel scheduling algorithms to families of perfect graphs, *Computing Suppl.*, **7**, 1990, 93-107.

[189] D. Kozen, U.V. Vazirani and V.V. Vazirani, NC Algorithms for comparability graphs, interval graphs, and testing for unique perfect matching, *Foundations of Software Technology and Theoretical Computer Science (New Delhi, 1985)*, Ed.: S.N. Maheshwari, Lecture Notes in Computer Science Vol. **206**, Springer-Verlag, Berlin, 1985, 496-503.

[190] M.O. Rabin and V.V. Vazirani, Maximum matchings in general graphs through randomization, *J. Algorithms*, **10**, 1989, 557-567.

[191] D. Yu. Grigoriev and M. Karpinski, The matching problem for bipartite graphs with polynomially bounded permanents is in NC (extended abstract), *Proc. 28th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1987, 166-172.

[192] A. Subramanian, The computational complexity of the circuit value and network stability problems, Stanford Univ. Dept. of Computer Sci. Report No. STAN-CS-90-1311, May, 1990, (Ph.D. Thesis).

[193] E.W. Mayr and A. Subramanian, The complexity of circuit value and network stability, Stanford Univ. Dept. of Computer Sci. Report No. STAN-CS-89-1278, August, 1989.

[194] _____, The complexity of circuit value and network stability, *Proc. Structure in Complexity Theory (4th Ann. IEEE Conf.)*, 1989, 114-123.

[195] R.E. Ladner, The circuit value problem is log space complete for P, *SIGACT News*, **7**, 1975, 18020.

[196] A. Subramanian, A new approach to stable matching problems, Stanford Univ. Dept. of Computer Science Tech. Report STAN-CS-89-1275, 1989.

[197] H. Narayanan, H. Saran and V.V. Vazirani, Fast parallel algorithms for matroid union, arboresences, and edge-disjoint spanning trees, preprint, January, 1991.

[198] T. Fischer, A.V. Goldberg and S. Plotkin, Approximating matchings in parallel, Stanford Univ. Dept. of Computer Sci. Report No. STAN-CS-91-1369, June, 1991.

[199] N. Alon, L. Babai and A. Itai, A fast and simple randomized parallel algorithm for the maximal independent set problem, *J. Algorithms*, **7**, 1986, 567-583.

[200] M. Goldberg and T. Spencer, A new parallel algorithm for the maximal independent set problem, *Proc. 28th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1987, 161-165.

[201] _____, A new parallel algorithm for the maximal independent set problem, *SIAM J. Comput.*, **18**, 1989, 419-427.

[202] _____, Constructing a maximal independent set in parallel, *SIAM J. Discr. Math.*, **2**, 1989, 322-328.

[203] M.K. Goldberg, Parallel algorithms for three graph problems, *Proc. Seventeenth Southeastern Conf. on Combinatorics, Graph Theory and Computing*, Eds.: F. Hoffman et al., Congress. Numer. 54, 1986, 111-121.

[204] P. Turán, On the theory of graphs, *Colloq. Math.*, **3**, 1954, 19-30.

[205] M. Goldberg and T. Spencer, An efficient algorithm that finds independent sets of guaranteed size, *Proc. First Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, Philadephia, 1990, 219-225.

[206] X. He, A nearly optimal parallel algorithm for constructing maximal independent sets in planar graphs, preprint, 1987.

[207] S. Khuller, Extending planar graph algorithms to $K_{3,3}$-free graphs, *Foundations of Software Technology and Theoretical Computer Science, (Pune, India, 1988)*, Eds.: K.V. Nori and S. Kumar, Lecture Notes in Computer Science Vol. 338, Springer-Verlag, Berlin, 1988, 67-79.

[208] N. Dadoun and D.G. Kirkpatrick, Parallel algorithms for fractional and maximal independent sets in planar graphs, *Discrete Appl. Math.*, **27**, 1990, 69-83.

[209] D. Helmbold and E. Mayr, Perfect graphs and parallel algorithms, *Proc. IEEE 1986 International Conference on Parallel Processing*, IEEE, New York, 1986, 853-860.

[210] A.A. Bertossi and M.A. Bonuccelli, Some parallel algorithms on interval graphs, *Discr. Appl. Math.*, **16**, 1987, 101-111.

[211] J. Naor, M. Naor and A.A. Schäffer, Fast parallel algorithms for chordal graphs, *Proc. 19th Annual ACM Symposium on Theory of Computing*, ACM, New York, 1987, 355-364.

[212] _____, Fast parallel algorithms for chordal graphs, *SIAM J. Comput.*, **18**, 1989,327-349.

[213] E. Dahlhaus and M. Karpinski, A fast parallel algorithm for computing all maximal cliques in a graph and the related problems (Extended Abstract), *SWAT 88 (Halmstad, Sweden, 1988)*, Lecture Notes in Computer Science Vol. **318**, 1988, 139-144.

[214] P. Kasteleyn, Dimer statistics and phase transitions, *J. Math. Phys.*, **4**, 1963, 287-293.

[215] _____, Graph theory and crystal physics, *Graph Theory and Theoretical Physics*, Ed.: F. Harary, Academic Press, New York, 1967, 43-110.

[216] C.H.C. Little, An extension of Kasteleyn's method of enumerating the 1-factors of planar graphs, *Combinatorial Mathematics, Proc. Second Australian Conference*, Ed.: D. Holton, Lecture Notes in Math., Vol. **403**, Springer-Verlag, Berlin, 1974, 63-72.

[217] L. Lovász, Matching structure and the matching lattice, *J. Combin. Theory Ser. B*, **43**, 1987, 187-222.

[218] V.V. Vazirani and M. Yannakakis, Pfaffian orientations, 0/1 permanents and even cycles in directed graphs, *Automata, Languages and Programming, (Tampere, Finland, 1988)*, Eds.: T. Lepistö and A. Salomaa, Lecture Notes in Computer Science Vol. **317**, Springer-Verlag, Berlin, 1988, 667-681.

[219] _____, Pfaffian orientations, 0–1 permanents, and even cycles in directed graphs, *Discrete Appl. Math.*, **25**, 1989, 179-190.

[220] W.R. Pulleyblank, personal communication, December, 1991.

[221] P. Seymour and C. Thomassen, Characterization of even directed graphs, *J. Combin. Theory Ser. B*, **42**, 1987, 36-45.

[222] M. Jerrum, Two-dimensional monomer-dimer systems are computationally intractable, *J. Stat. Phys.*, **48**, 1987, 121-134. (Erratum: *J. Stat. Phys.*, **59**, 1990, 1087-1088.)

[223] A. Sinclair, Randomised algorithms for counting and generating combinatorial structures, Univ. Edinburgh Dept. of Computer Science Report CST-58-88, Ph.D. Thesis, November, 1988.

[224] R.W. Irving and P. Leather, The complexity of counting stable marriages, *SIAM J. Comput.*, **15**, 1986, 655-667.

[225] H. Minc, *Permanents*, Encyclopedia of Mathematics and its Applications, **6**, Addison-Wesley, Reading, 1978.

[226] H. Ryser, *Combinatorial Mathematics*, Carus Mathematical Monograph No. 14, Math. Assoc. Amer., Washington, 1963.

[227] M. Luby, A survey of approximation algorithms for the permanent, *Sequences (Naples/Positano, 1988)*, Ed.: R.M. Capocelli, Springer-Verlag, New York, 1990, 75-91.

[228] N. Karmarkar, R. Karp, R. Lipton, L. Lovász and M. Luby, A Monte-Carlo algorithm for estimating the permanent, preprint, May 5, 1988.

[229] M. Jerrum and U.V. Vazirani, A mildly exponential approximation algorithm for the permanent, preprint, September, 1991.

[230] A.Z. Broder, How hard is it to marry at random? (On the approximation of the permanent), (Extended Abstract), *Proc. 18th Annual ACM Sympos. on Theory of Computing (Berkeley, Calif.)*, ACM, New York, 1986, 50-58. Also see errata: *Proc. 20th Annual ACM Sympos. on Theory of Computing (Chicago, Ill.)*, ACM, New York, 1988, 551.

[231] M. Jerrum and A. Sinclair, Conductance and the rapid mixing property for Markov chains: the approximation of the permanent resolved (Preliminary Version), *Proc. 20th ACM Symposium on Theory of Computing*, ACM, New York, 1988, 235-244.

[232] _____, Approximating the permanent, *SIAM J. Comput.*, **18**, 1989, 1149-1178.

[233] P. Dagum, M. Luby, M. Mihail and U.V. Vazirani, Polytopes, permanents and graphs with large factors, *Proc. 29th IEEE Symposium on Foundations of Computer Science (White Plains)*, IEEE, Computer Society Press, 1988, 412-421.

[234] _____, Polytopes, permanents and graphs with large factors, *Theoret. Comput. Sci.*, 1991, (to appear).

[235] P. Dagum and M. Luby, Approximating the premanent of graphs with large factors, preprint, February, 1991.

[236] K. Fukuda and T. Matsui, Finding all the perfect matchings in bipartite graphs, Tokyo Inst. Tech. Dept. of Information Sciences, Res. Rep. Inform. Sci. Ser. B: Operations Research, Report B-225, September, 1989.

[237] N. Chiba and T. Nishizeki, Arboricity and subgraph listing algorithms, *SIAM J. Comput.*, **14**, 1985, 210-223.

[238] S. Tsukiyama, M. Ide, H. Ariyoshi and I. Shirakawa, A new algorithm for generating all maximal independent sets, *SIAM J. Comput.*, **6**, 1977, 505-517.

[239] L. Lovász, The work of A.A. Razborov, International Congress of Mathematicians, Kyoto, August, 1990.

[240] A.A. Razborov, Lower bounds on the monotone circuit complexity of some Boolean functions, *Doklady Akad. Nauk SSSR*, **281**, 1985, 798-801. (Russian). (English transl.: *Soviet Math. Dokl.*, **31**, 1985, 354-357.)

[241] _____, Lower bounds on monotone circuit complexity of the logical permanent, *Mat. Zametki*, **37**, 1985, 887-900. (Russian). (English transl.: *Math. Notes of the Acad. Sciences of USSR*, **37**, 1985, 485-493.)

[242] N. Alon and R.B. Boppana, The monotone circuit complexity of Boolean functions, *Combinatorica*, **7**, 1987, 1-22.

[243] H.N. Gabow, Scaling algorithms for network problems, *Proc. 24th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1983, 248-257.

[244] _____, A scaling algorithm for weighted matching on general graphs, *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1985, 90-100.

[245] _____, Data structures for weighted matching and nearest common ancestors with linking, *Proc. First Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, Philadelphia, 1990, 434-443.

[246] H.N. Gabow, Z. Galil and T.H. Spencer, Efficient implementation of graph algorithms using contraction, *Proc. 25th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1984, 347-357.

[247] _____, Efficient implementation of graph algorithms using contraction, *J. Assoc. Comput. Mach.*, **36**, 1989, 540-572.

[248] R. Anstee, A polynomial algorithm for b-matchings: an alternative approach, Univ. British Columbia Dept. Math. Preprint, 1986.

[249] N. Calkin and A. Frieze, Probabilistic analysis of a parallel algorithm for finding maximal independent sets, Carnegie Mellon Univ. Dept. of Math. Research Report No. 88-38, February, 1990.

[250] D. Coppersmith, P. Raghavan and M. Tompa, Parallel graph algorithms that are efficient on average, *Proc. 28th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1987, 260-269.

[251] A.M. Frieze, Probabilistic analysis of graph algorithms, Carnegie Mellon Univ. Research Report No. 88-42, February, 1989.

[252] _____, Probabilistic analysis of graph algorithms, *Computing, Supp.*, **7**, 1990, 209-233.

[253] M. Jerrum, An analysis of a Monte Carlo algorithm for estimating the permanent, preprint, June, 1991.

[254] A.M. Frieze, Maximum matchings in a class of random graphs, *J. Combin. Theory Ser. B*, **40**, 1986, 196-212.

[255] _____, On matchings and Hamilton cycles in random graphs, Carnegie Mellon Univ. Dept. of Math. Research Report No. 88-36, October, 1988.

[256] _____, On perfect matchings in random bipartite graphs with minimum degree at least two, Carnegie Mellon Univ. Dept. Math. Res. Rpt. No. 90-86, June, 1990.

[257] A.M. Frieze and D. Tygar, Deterministic parallel algorithms for matchings in random graphs, preprint, 1990, in preparation.

[258] O. Goldschmidt and D.S. Hochbaum, A fast perfect matching algorithm in random graphs, *SIAM J. Discr. Math.*, **3**, 1990, 48-57.

[259] G.R. Grimmett, Random near-regular graphs and the node packing problem, *Oper. Res. Letters*, **4**, 1985, 169-174.

[260] G.R. Grimmett and W.R. Pulleyblank, An exact threshold theorem for random graphs and the nodepacking problem, *J. Combin. Theory Ser. B*, **40**, 1986, 187-195.

[261] M. Jerrum, The elusiveness of large cliques in a random graph, Univ. Edinburgh, Dept. of Computer Sci. Internal Report CSR-9-90, September, 1990.

[262] R. Motwani, Expanding graphs and the average-case analysis of algorithms for matchings and related problems, *Proc. 21st Annual ACM Symposium on Theory of Computing*, ACM, New York, 1989, 550-561.

[263] A.V. Goldberg, S.A. Plotkin, D.B. Shmoys and É. Tardos, Using interior point methods for fast parallel algorithms for bipartite matching and related problems, preprint, March, 1991.

[264] D.S. Johnson, C. Papadimitriou and M. Yannakakis, On generating all maximal independent sets, *Inform. Proc. Lett.*, **27**, 1988, 119-123.

A Complexity Class Framework